PREFACE

This is the user's manual for ESA 86/88-2 microprocessor trainer. This manual describes the hardware and software components of ESA 86/88-2 and gives the interface information necessary for expanding the system.

This manual describes in detail the features offered by the Keyboard monitor program, Serial monitor program and the one line Assembler / Disassembler package. The optional features; Audio cassette interface, EPROM programmer Interface, Centronics parallel printer interface are also described in this manual. Complete schematics and related drawings are provided in the appendices.

Please note that this volume is a user's guide for ESA 86/88-2 and as such does not deal elaborately with the features of 8086/8088 processor and related peripherals and their programming. Details regarding these can be obtained from the following INTEL publications.

iAPX 86/186 USER'S MANUAL
8086 FAMILY USER'S MANUAL
MICRO SYSTEMS COMPONENT HANDBOOK VOL I AND II

While every effort has been made to present the information in an accurate and simple fashion, we do welcome suggestions for improving the quality and usefulness of this manual.

Please address all your correspondence to :

ELECTRO SYSTEMS ASSOCIATES PVT LTD
4215 J K COMPLEX FIRST MAIN ROAD
SUBRAMANYANAGAR PO BOX 2139
BANGALORE - 560 021 INDIA
Phone : 3322924 3323029
Fax : 080 – 3325615, e-mail : esaindia@.vsnl.com

CONTENTS

	PAGE
CHAPTER 1 INTRODUCTION	**1.1**
CHAPTER 2 CONFIGURATION AND INSTALLATION	**2.1**
2.1 Configuration of ESA 86/88-2	2.1
2.1.1 Processor selection	2.1
2.1.2 Operational mode selection	2.1
2.1.3 Printer Enable/Disable	2.1
2.1.4 Baud rate selection	2.2
2.1.5 Memory selection	2.2
2.1.6 Interrupt selection	2.3
2.1.7 8087 installation	2.3
2.2 Installation of ESA 86/88-2	2.3
2.2.1 Installation procedure for Serial mode of operation	2.4
2.2.2 No response in Serial mode	2.4
2.2.3 Installation procedure for Keyboard mode of operation	2.5
2.2.4 No response in Keyboard mode	2.5
CHAPTER 3 KEYBOARD MONITOR	**3.1**
3.1 Introduction	3.1
3.2 General operation	3.1
3.3 Monitor Commands	3.2
3.4 Examine Byte and Examine Word Commands	3.5
3.5 Examine Register command	3.9
3.6 Input / Output Commands	3.11
3.6.1 Input byte and Input word commands 3.11	
3.6.2 Output byte and output word commands	3.12
3.7 Step Command	3.14
3.8 Go Command	3.15
3.9 Move Command	3.16
3.10 Use of REG, + and - keys	3.19
3.10.1 Use of “+” key	3.19
3.10.2 Use of “-” key	3.19
3.10.3 Use of "REG" key	3.20
3.11 User programmable function keys	3.21
CHAPTER 4 SERIAL MONITOR	**4.1**
4.1 Introduction	4.1
4.2 General operation	4.1
4.3 Structure of monitor commands	4.2
4.3.1 Parameter entry	4.2
4.3.2 Use of +,- operators and register, in specification of Parameters	4.2
4.3.3 Response to Errors	4.3
4.4 Monitor commands	4.3
4.4.1 Substitute Memory commands	4.5
4.4.2 Display memory commands	4.6
4.4.3 Examine/modify Register command	4.7
4.4.4 Move command	4.8
4.4.5 Input Byte and Input word commands	4.10

4.4.6 Output Byte and Output word commands 4.11

4.4.7 Go command 4.12
4.4.7.1 Break point facility 4.12
4.4.8 Single step command 4.14
4.4.9 Read HEX file command 4.14
4.4.10 Write HEX file command 4.15

CHAPTER 5 HARDWARE 5.1
5.1 CPU 5.1
5.1.1 CPU Reset 5.1
5.1.2 CPU Clock 5.1
5.1.3 CPU Address Bus 5.1
5.1.4 Corrected BHE* 5.1
5.1.5 CPU Data Bus 5.2
5.2 Memory addressing 5.2
5.2.1 Memory map 5.3
5.3 Peripheral clock 5.3
5.4 I/O addressing and I/O map 5.3
5.4.1 Parallel I/O ports 5.3
5.4.2 Onboard Keyboard/Display 5.4
5.4.3 Programmable interval timer 5.5
5.4.4 Serial interface controller 5.6
5.4.5 Programmable interrupt controller 5.6
5.4.6 I/O address map 5.6
5.5 Wait state logic 5.7
5.6 Interrupt system 5.7
5.6.1 Hardware Interrupts 5.7
5.6.2 Internal Interrupts 5.7
5.7 8087 Numeric Data Processor 5.7
5.8 Bus expansion 5.7
5.9 Connector details 5.8

CHAPTER 6 MONITOR ROUTINES ACCESSIBLE TO USER 6.1
6.1 Keyboard Monitor routines accessible to user 6.1
6.2 Serial Monitor routines accessible to user 6.3

CHAPTER 7 ESAM-86 SYMBOLIC ASSEMBLER 7.1
7.1 Introduction 7.1
7.1.1 Installation 7.1
7.2 Operation 7.1
7.2.1 Invoking the single line Assembler ESAM-86
7.1
7.2.2 Exiting from ESAM-86 7.2
7.2.3 Structure of ESAM-86 Commands 7.2
7.2.4 Description of ESAM-86 Commands 7.3
7.2.4.1 SG Command 7.3
7.2.4.2 LC Command 7.3
7.2.4.3 DA Command 7.3
7.2.4.4 DI Command 7.5
7.3 Assembly language conventions 7.5
7.3.1 Label field 7.6
7.3.2 Mnemonic field 7.6

7.3.2.1 Segment override option 7.6
7.3.2.2 Directives (Pseudo opcodes) 7.7
7.3.3 Operand field 7.9
7.3.4 Addressing modes 7.9
7.3.5 Mnemonic Syntax 7.10

CHAPTER 8 PROGRAMMING EXAMPLES 8.1
8.1 Introduction 8.1
8.2 Familiarization examples 8.1
8.3 Use of Monitor routines 8.5
8.4 Use of 8087 Co-processor 8.11
8.5 Use of programmable interrupt Controller-8259A 8.15

CHAPTER 9 AUDIO TAPE INTERFACE 9.1
9.1 Introduction 9.1
9.2 Installation 9.1
9.3 Operation from Serial Monitor 9.1
9.3.1 T Command : Transfer from memory to tape 9.1
9.3.2 L Command : Load from tape 9.3
9.3.3 Response to entry errors 9.5
9.4 Operation from Keyboard Monitor 9.5
9.4.1 TWR Command : Transfer from memory to tape 9.5
9.4.2 TRD Command : Load into memory from tape 9.6
9.4.3 Response to entry errors 9.8
9.5 General hints 9.8
9.6 Theory of operation 9.8
9.6.1 Data format 9.8
9.6.2 Storage capacity 9.9

CHAPTER 10 PROM PROGRAMMER SYSTEM 10.1
10.1 Introduction 10.1
10.2 Installation 10.1
10.3 Operation from Serial Monitor 10.2
10.3.1 P Command 10.3
10.3.2 V Command 10.5
10.3.3 B Command 10.5
10.3.4 R Command 10.6
10.3.5 E Command 10.7
10.4 Operation from keyboard monitor 10.7
10.4.1 PROG Command 10.8
10.4.2 Verify Command 10.9
10.4.3 Blnk Chk Command 10.9
10.4.4 TRFER Command 10.10
10.5 Examples 10.11

CHAPTER 11 PARALLEL PRINTER INTERFACE 11.1
11.1 Introduction 11.1
11.2 Installation and operation 11.1
11.3 Theory of operation 11.2
11.4 Error messages 11.2
11.5 Parallel printer port details 11.2
11.6 Connector details 11.3
11.7 Direct output to printer 11.3

CHAPTER 12 COMMUNICATION WITH A HOST COMPUTER SYSTEM **12.1**
12.1 Introduction 12.1
12.2 Installation 12.1
12.3 Returning to DOS 12.3
12.4 Operational details 12.3
12.4.1 Download operation 12.3
12.4.2 Upload operation 12.4
12.4.3 DOS commands 12.5
12.4.4 Status Line 12.6
12.4.5 Command recall 12.6
12.4.6 Communication 12.6
12.4.7 Help 12.6
12.5 More about the ".HEX" file 12.7
12.5.1 Intel extended HEX format 12.7
12.5.2 Using X8086 12.9
12.5.3 Downloading .EXE files 12.10
12.5.4 Downloading .COM files 12.12
12.6 Demo files 12.14

APPENDICES

APPENDIX A - SCHEMATICS
APPENDIX B - CONNECTOR DETAILS
APPENDIX C - COMPONENT LAYOUT
APPENDIX D - ASCII CHARACTER SET
APPENDIX E - RS 232 C CABLE DETAILS
APPENDIX F - 8087 INSTRUCTION SET
APPENDIX G - PRODUCT LIST

CHAPTER 1
INTRODUCTION

ESA 86/88-2 is a powerful, general purpose microcomputer system which can be operated either with 8086 CPU or with 8088 CPU. It is generally supplied with 8086 CPU. To change it to 8088, user has to just remove the 8086, insert 8088 into that socket and set a DIP switch.

The 8086 and 8088 are third generation CPUs from INTEL that differ primarily in their external data paths. 8088 uses an 8-bit wide data bus while 8086 uses a 16-bit wide data bus. ESA 86/88-2 can be operated with either CPU and the only possible difference would be in the speed of execution (with 8088 CPU, a small speed degradation occurs because of the 8-bit wide data bus). In either case, the CPU is operated in the maximum mode.

ESA 86/88-2 can be used not only as an instructional and learning aid, but also as a development tool in R&D labs and industries. It is a versatile system and can be configured in a variety of ways, as per the user requirements, via various jumper options. The basic system can be easily expanded through the system BUS connectors. The built-in Assembler/Disassembler feature simplifies the programmer's task of entering Assembly language programs. The On-board provision for 8087 Numeric Data Processor makes ESA 86/88-2 useful for number-crunching applications also. Onboard battery backup provision is an added feature to take care of the frequent power failures while conducting experiments on the trainer using manually assembled code.

To aid the user further various optional attachments like EPROM Programmer, Audio cassette interface, are available, details of which are also explained in this manual. ESA 86/88-2 is operated from the C.R.T terminal or a host computer system via the serial monitor and also can be operated from the onboard keyboard.

Following are the system capabilities

- Examine and optionally modify the contents of memory (byte or word format).
- Examine and optionally modify the processor registers.
- Assemble and Disassemble 8086/8088 instructions (via line assembler, disassembler).
- Perform fast numerical computations using the optional 8087 Numeric Data Processor.
- Execute the user program at full speed.
- Debug user program through Single Step and Breakpoint facilities.
- Write or read data to or from I/O ports (byte or word format).
- Move a block of data or program within the memory
- Download user programs into ESA 86/88-2 from a host computer system.
- Transfer program code or data to and from Audio Cassette (via optional audio cassette interface module)
- Read, Program, Verify and Blank-check an EPROM (via optional PROM programmer module).
- Obtain a hard copy (via centronics compatible parallel printer interface).

SPECIFICATIONS:

Central Processor

8086 CPU or 8088 CPU, operating at 8 MHz in maximum mode.
(Supplied with 8086 CPU).
(Memory cycles have zero wait states and I/O cycles have one wait state)

Co-Processor

On-board 8087 Numeric Data processor (optional)

Memory

EPROM: 4 JEDEC compatible slots offer the following options:

64K bytes using 27128s or,
128K bytes using 27256s or,
256K bytes using 27512s

(system firmware is supplied in 2X27256s. The other two sockets are for user expansion).

RAM : 4 JEDEC Compatible slots offer the following:

128K bytes using 62256s
(64K Bytes supplied using 2x 62256s. The other two sockets are for user expansion). RAM has battery backup facility.

Peripherals and Controllers

8251A : Programmable Communication Interface for serial communication supporting all standard bauds from 110- 19,200 (Baud is selected through on-board DIP switch).

8253-5: Programmable Interval timer; Timer 0 is used for generating baud clock. Timer 1 and Timer 2 are available for user.

8255A: (2 Nos.) Programmable Peripheral Interface devices provide 48 Programmable I/O lines.

8259A: Programmable Interrupt Controller provides interrupt vectors for 8 sources

8288: Bus Controller used for generating control signals.

Interrupts

External: NMI for INTR key of Keyboard.
INTR controlled through 8259A,on-board Interrupt Controller: provides interrupt vectors for eight sources. Complete flexibility in selecting either off-board or on-board interrupt sources.

	On-board interrupt sources	
#	8251	(TxRDY and RxRDY)
#	8253-5	(OUT1 and OUT2)
#	8255A	(PC0 and PC3 in Handshake Mode)
#	8087	(NDP INT)

Internal: Interrupt Vectors 1 (single step) and 3 (breakpoint) reserved for monitor.

Interface Signals

CPU Bus **:** Demultiplexed and fully buffered, TTL compatible, Address, Data & Control signals are available on two 50 pin ribbon cable connectors.

Parallel I/O **:** 48 programmable parallel I/O lines (TTL Compatible) through two 26 pin ribbon cable connectors. (Connector details compatible to our other microcomputer trainers).

Serial I/O **:** RS 232 C through on-board 9 pin D-type female connector.

Power Supply (Optional) **:** + 5V @ 3.0 Amp

CHAPTER 2

CONFIGURATION AND INSTALLATION

2.1 Configuration of ESA 86/88-2

ESA 86/88-2 microcomputer trainer is versatile and can be configured in a number of ways, as determined by the settings of a DIP switch and other jumpers. (Refer to the component layout diagram in Appendix C to locate the DIP switch and the jumpers). This chapter describes all the configuration options and the installation procedures.

2.1.1 Processor selection

ESA 86/88-2 can be operated either with an 8086 CPU or with an 8088 CPU. Install the desired CPU (8086 or 8088) in the socket position U29. Then set the SW8 of the DIP switch as shown below:

SW8 of DIP Switch	Processor
OFF	8086 *
ON	8088

(* Factory installed option)

2.1.2 Operational mode selection

ESA 86/88-2 can be operated either in the serial mode or in Hexadecimal keypad mode. In the serial mode, the trainer is connected to a CRT terminal or to a host computer system (like PC compatible) through an RS 232 C interface. In the keypad mode, the trainer is operated through Hexadecimal Keypad. The selection of the desired mode of operation is done as follows:

SW4 of the DIP Switch	Operational mode
OFF	Serial mode
ON	Hexadecimal keypad mode*

(* Factory installed Option)

Chapter 3 describes the commands available in keypad mode and chapter 4 describes the commands available in the serial mode.

2.1.3 Printer Enable/Disable

ESA 86/88-2 firmware includes the driver program for centronics compatible parallel printer interface. This driver can be enabled/disabled as shown below:

SW5 of the DIP Switch	Printer Driver
OFF	Disabled *
ON	Enabled

(* Factory installed Option)

Chapter 11 describes the interface in detail.

2.1.4 Baud rate selection

In the serial mode of operation, ESA 86/88-2 configures an 8251A USART as follows:

- Asynchronous mode
- 8-bit character length
- 2 stop bits
- No parity
- Baud rate factor of 16X

Timer 0 of an 8253 provides the Transmit and Receive baud clocks for the USART. (Refer to chapter 5 for a detailed discussion of the Hardware). This timer is initialized by the system firmware to provide proper baud clock based on the settings of the DIP Switch as shown below.

DIP SWITCH			Baud rate
SW3	SW2	SW1	
ON	ON	ON	110
ON	ON	OFF	300
ON	OFF	ON	600
ON	OFF	OFF	1,200
OFF	ON	ON	2,400
OFF	ON	OFF	4,800
OFF	OFF	ON	9,600 *
OFF	OFF	OFF	19,200

(* Factory installation option)

2.1.5 Memory Selection

ESA 86/88-2 has four 28-pin sockets, labeled U9, U8, U7, U6 for RAM. These sockets are configured for 62256 (32K X 4) devices. Two of these sockets are populated (providing 64K Bytes of RAM) and two are for user expansion.

Four 28-pin sockets; U5, U4, U3, U2 are dedicated to EPROM. These four sockets can be collectively configured for 27128, 27256 or 27512 as shown below:

DEVICE	DIP SWITCH		JUMPER
	SW7	SW6	
27128	ON	ON	JP1 - 2-3
			JP9 - 2-3
27256	ON	OFF	JP10 - 1-2*
			JP9 - 2-3
27512	OFF	OFF	JP1 - 1-2
			JP9 - 1-2

(* Factory installed option. System firmware is supplied in 2 X 27256 EPROMs, at U3 and U2. The other two sockets U5 and U4 are for user expansion).

2.1.6. Interrupt Selection

ESA 86/88-2 features an 8259A Priority Interrupt controller. The interrupt sources to 8259A can be selected to be either on-board signals or off-board signals. The selection is as follows:

JUMPER	INTERRUPT SELECTED	8259A INTERRUPT
JP8 – 2-3	PC3 of U32 (8255A)	
JP8 – 1-2	INT0*	IR0
JP7 – 2-3	PC3 of U27 (8255A)	
JP7 – 1-2	INT1*	IR1
JP6 – 2-3	PC0 of U27 (8255A)	
JP6 – 1-2	INT2*	IR2
JP5 – 2-3	TxRDY of U13 (8251A)	
JP5 – 1-2	INT3*	IR3
JP4 – 2-3	RxRDY of U13 (8251A)	
JP4 – 1-2	INT4*	IR4
JP3 – 2-3	TIMER 1 of U12 (8253)	
JP3 - 1-2	INT5*	IR5
JP2 – 2-3	TIMER 2 of U12 (8253)	
JP2 – 1-2	INT6*	IR6
JP1 – 2-3	NDPINT of U28 (8087)	
JP1 – 1-2	INT7*	IR7

(INT0* to INT7* are from System connector J1 Factory installation leaves all the jumpers open)

2.1.7. 8087 installation:

ESA 86/88-2 has on-board provision for 8087 Numeric Data Processor. To install it, insert the 8087 into the socket labelled U28. No other jumpers or hardware changes are necessary.

2.2 Installation of ESA 86/88-2

To install ESA 86/88-2, the following accessories are required

a) Power Supply
5V, 3.0 Amp

30V @ 100mA, 12V @ 250Ma if PROM Programmer interface is being used. (Refer chapter 10 for details of this interface)

b) For Serial mode of operation :

CRT terminal with RS 232 C interface OR Host system (like PC Compatible) with the driver software for host system. (Refer chapter 12 for details)

2.2.1. Installation procedure for Serial mode of operation :

a) Select the processor 8086 or 8088 (Ref. Section 2.1.1)
b) Select Serial mode of operation (Ref. Section 2.1.2)
c) Select printer if required (Ref. Section 2.1.3)
d) Set the desired baud rate (Ref. Section 2.1.4)
e) Select interrupt sources if required (Ref. Section 2.1.6)
f) Select EPROM configuration if necessary (Ref.Section 2.1.5)
g) Install 8087 NDP if desired (Ref. Section 2.1.7)
h) Connect ESA 86/88-2 to the CRT terminal / Host system through an RS 232 C cable (Appendix E describes the RS 232 C interface requirements) over the connector J3. (Refer Appendix C for locating the connectors).

If a terminal is being used, turn on the terminal. If a computer system is being used, turn on the system and execute the driver program. (Ref. Chapter 12 for details)

i) Connect the Power Supply of required capacity to ESA 86/88-2 and turn on the power.
j) Press the RESET Key on ESA 86/88-2.

Now the following sign-on message should appear on the screen if 8086 has been installed.

ESA-86 Serial monitor V x.y

If 8088 has been installed, the sign-on message should be

ESA-88 Serial monitor V x.y

(V x.y indicates version x and revision y)

The sign-on message is followed by the command prompt, "." in the next line.
Now ESA 86/88-2 is ready for operation in Serial mode.

NOTE : 8 Digit seven segment Display on ESA 86/88-2 will display "E Se 86" or "E Se 88".

2.2.2 NO Response in Serial mode

If there is no response from ESA 86/88-2 in Serial mode, after installing it as described in the previous section, check the following items:

a) Check the RS 232 C cable connections at both the ends. (Appendix E describes the interface in detail)
b) Check the power supply connections and voltage levels.
c) Check the handshake signals of RS 232 C interface (Ref. Appendix E)
d) Check the baud rates of ESA 86/88-2 and the device connected to it.
e) If a computer system is the controlling device, check that the driver program is running, the RS 232 C cable is connected to the correct port and that the port is working
f) Check the configuration of ESA 86/88-2 again. (DIP Switch settings, jumpers).

NOTE: DIP Switch status is read only at power ON / reset. If you change the settings, either press the RESET key or switch off and then switch on the power supply.

If the problem still persists, please contact the manufacturer.

2.2.3. Installation procedure for keyboard mode of operation.

a) Select the processor 8086 or 8088 (Ref. Section 2.1.1)
b) Select Keyboard mode of operation (Ref. Section 2.1.2)
c) Select printer if required (Ref. Section 2.1.3)
d) Select interrupt sources if required (Ref. Section 2.1.6)
e) Set the EPROM configuration if necessary (Ref. Section 2.1.5)
f) Install the 8087 NDP if desired (Ref. Section 2.1.7).
g) Connect the power supply of required capacity to ESA 86/88-2 and switch on the power.
h) Press the RESET key.

Now if 8086 is installed, the following sign-on message will appear on the seven-segment display.

- ESA 86

If 8088 is installed, the sign-on message should be

- ESA 88

Now ESA 86/88-2 is ready for operation in the keyboard mode.

2.2.4. No response in Keyboard mode.

If the correct sign-on message does not appear in the keyboard mode, check the following items.

a) If the seven-segment display is totally blank, check the power supply connections.
b) If the seven-segment display shows random pattern, check the configuration settings once again.

NOTE: DIP Switch is read only at power ON / reset. If you change the settings, either press the RESET key or switch off and then switch on the power supply.

If the problem still persists, please contact the manufacturer.

CHAPTER 3

KEYBOARD MONITOR

3.1 INTRODUCTION

This chapter describes the user interaction with the ESA-86/88-2 through the use of keyboard monitor program.

With the keyboard monitor program, both commands and data are entered by pressing the individual keys on the keyboard of the system. Most of the keys have combined functions as indicated by their individual legends.

The system uses an eight-digit 7-segment LED display to communicate with the user. This display is divided into two groups of four characters each. The group on the left, generally used for the display of address values, is referred to as the "address field" and the group on the right, generally used for the display of data values is referred to as the "data field".

The keyboard/display interface to the system is described in detail in chapter 5.

This chapter is described assuming that the CPU installed is 8086. However, all the information provided here is valid without any changes even if 8088 CPU is installed. The only observable difference would be in the sign-on message displayed following a power ON/RESET.

3.2 GENERAL OPERATION

Following power-ON or whenever the Trainer RESET key is pressed, the monitor initializes the ESA 86/88-2 and displays the monitor sign-on message, "ESA 86" (if 8086 is installed; "ESA 88" if 8088 CPU is installed) and the command prompt character ("-") in the most significant digit of the address field. Whenever the Trainer RESET key is pressed, all information about any previous user program is lost. However, the contents of the user portion of the RAM are not disturbed. Initially the 8086/8088 registers are set to the values shown in Table 3.1

Table 3.1 Register Initialization

Register	Value (Hex.)
CS (Code Segment)	0
DS (Data Segment)	0
ES (Extra Segment)	0
SS (Stack Segment)	0
IP (Instruction Pointer)	0
FL (Flags)	0
SP (Stack Pointer)	100

Further, interrupt vectors 1, 2 and 3 are initialized as follows :

Interrupt 1 : Single Step-used with the step command
Interrupt 2 : NMI (Non-Maskable Interrupt)-Monitor's INTR key
Interrupt 3 : Breakpoint--used with the GO command.

A detailed discussion of the 8086/8088 interrupt vectors can be found in the Intel's 8086 Family User's Manual (Chapter 2, Processor Control and Monitoring, Interrupts PP 2.22-2.28).

Whenever the monitor is re-entered as a result of a single step, NMI or Breakpoint Interrupt, the monitor saves the contents of the 8086/8088 registers on the User Stack and subsequently restores the register contents from the stack before it prompts for command entry. Since the SP register is initialized to 0100H, and memory locations 0H-CFH are reserved for monitor, the stack length reserved for the user is 48 bytes (location D0H - FFH): of these 26 bytes must be left for register contents, should one of the above interrupts occur (leaving 22 bytes for the user). In addition locations 100H to 1FFFH are used for other system functions and system tables.

When using the keyboard monitor, user is prompted (through the display) for the inputs required. Whenever the monitor is expecting a command entry, a dash ("-") appears in the left-most digit position of the display. When the monitor expects the entry of command parameters, one or more dots (depending upon the number of parameters required) appear in one of the two display fields (address field and data field) depending upon the parameter required. Also depending upon the command, other characters may appear in the address and data fields.

Whenever an error is detected by the monitor (either in the command entry or in command execution), the characters "Err" are displayed in the address field along with the command prompt character ("-").

ENTRY OF ADDRESS AND DATA VALUES

When a command requires the entry of an address/data value from the user, the following rules apply.

All addresses in the 8086/8088 system consist of a segment value and an offset value. The segment value is entered first, a colon (the key`:') is entered as a separator, and then the offset value is entered. If the segment value is not specified (note that, in such a case, the colon also should be omitted), the default segment value is the current contents of the code segment (CS) register. Enter the address as Hexadecimal value, most significant character first. The capacity of an address field entry is limited to four characters and if more than four characters are entered (for either segment or offset value) only the last four characters entered (the four characters currently displayed) are valid. In otherwords, all address values are interpreted modulo 64K. Please note that an address value can be specified as the contents of any of the 8086/8088's registers also using the "Reg" key and appropriate key identifying the specific register.

The segment value (either entered by the user or the default CS value) is shifted left by four bits and added to the offset value to form the 20-bit physical address.

Data is also entered as hexadecimal value, most significant character first. The data field is limited to
either two characters (for byte values) or four characters (for word values). If more characters are entered, only the characters currently displayed in the field are valid. Thus byte values are treated modulo 256 and word values are treated modulo 64K. Data values also can be specified as the contents of an appropriate 8086/8088 register.

3.3 MONITOR COMMANDS:

Ten individual commands can be executed using the keyboard monitor. These commands are summarized in Table 3.4. Individual commands are described in detail in the sections which follow.

In both the table and the individual command descriptions, the following notation is used.

[X] Indicates a keyboard key
[V] Indicates that "V" is optional
[V]* Indicates one or more optional occurrences of "V"
<V> Indicates that "V" is a parameter to be entered by the user.

Table 3.2 Hexadecimal Keypad Legend Interpretation

Hexadecimal key	Command Acronym	Command Name	Register Acronym	Register Name
0 EB/AX	EB	Examine Byte	AX	Accumulator
1 ER/BX	ER	Examine Register	BX	Base Register
2 GO/CX	GO	Go	CX	Count Register
3 ST/DX	ST	Single Step	DX	Data Register
4 IB/SP	IB	Input Byte	SP	Stack pointer
5 OB/BP	OB	Output Byte	BP	Base Pointer
6 MV/SI	MV	Move	SI	Source Index
7 EW/DI	EW	Examine Word	DI	Destination index
8 IW/CS	IW	Input Word	CS	Code Segment
9 OW/DS	OW	Output Word	DS	Data Segment
A /SS	none	N/A	SS	Stack Segment
B /ES	none	N/A	ES	Extra Segment
C /IP	none	N/A	IP	Instruction Pointer
D /FL	none	N/A	FL	Flag Register
E	none	N/A	none	N/A
F	none	N/A	none	N/A

Table 3.3 Function Key Operation

Function Key	Operation
RESET	The RESET key allows you to terminate any present activity and to return your ESA 86/88-2 to an initialized state. When pressed, the sign-on message appears in the display and the monitor is ready for command entry.
KB INT	The INTR (interrupt) key is used to generate an immediate, non-maskable type 2 interrupt (NMI). The NMI interrupt vector is initialized on power up or system reset to point to a routine within the monitor which causes all of the 8086/8088's registers to be saved. Control is returned to the monitor for subsequent command entry.
+	The + (plus) key allows you to add two hexadecimal values. This function simplifies relative addressing by allowing you to readily calculate an address location relative to a base address.
-	The - (minus) key allows you to subtract one hexadecimal value from another.
:	The : (colon) key is used to separate an address to be entered into two parts; a segment value and an offset value.
REG	The REG (register) key allows you to use the contents of any of the 8086/8088's registers as an address or data value.
NEXT	The NEXT key is used to separate keypad entries and
(,)	to increment the address field to the next consecutive memory location.
PREV	The PREV key is used to decrement the address field to previous memory location.
EXEC	The dot Key is the command terminator. When
(.)	pressed, the current command is executed.

NOTE: 1) NEXT or , means the same operation
2) EXEC or . means the same operation

TABLE 3.4 Summary of Monitor commands

Command Group	Command	Function/Format
Examine/modify	Examine Byte	Displays/modifies memory byte locations **EB <address>NEXT [[<data>] NEXT or PREV]*.**
	Examine Word	Displays/modifies memory Word locations **EW <address>, [[<data>] ,] NEXT or PREV]* .**
	Examine Register	Displays / modifies processor register contents **ER <reg key> [[<data>] NEXT] [.]**
Input/Output	Input Byte	Displays the data byte at the input port. **IB <port address> NEXT [NEXT]* .**
	Input word	Displays the data word at the input port. **IW <port address> NEXT [NEXT]*.**
	Output Byte	Outputs data byte to the output port **OB <port address> NEXT <data> [NEXT <data>] * .**
	Output Word	Outputs data word to the output port. **OW <port address> NEXT <data> [NEXT <data>]*.**
Execution	Step	Executes one single instruction. **ST [<Start address>]NEXT [[<start address>]NEXT] * .**
	Go	Transfers control from monitor to user program **GO [<address>] [NEXT<breakpoint address>].**
Block Move	Move	Moves block of data within memory **MV <start address> NEXT <end address> NEXT <destination address>** .

3.4 EXAMINE BYTE AND EXAMINE WORD COMMANDS

Function: The Examine Byte (EB) and Examine Word (EW) commands are used to examine the contents of selected memory locations. If the memory location can be modified (e.g. a location in RAM), the contents can optionally be modified.

Format

EB <address> NEXT [[<data>] PREV/NEXT] * .

EW <address> NEXT [[<data>] PREV/NEXT] * .

Operation

1. Both the commands operate in a similar fashion. To use these comnmands, press the EB key or EW key when prompted for a command.

2. When either key is pressed, a dot appears at the right edge of the address field indicating that an address entry is required.

3. Enter the memory address of the byte (for EB) or word (for EW) to be examined. (The entry of address values is discussed in detail in section 3.2)

4. After entering the address value, press the "," key. (i.e. the NEXT key).

5. The data byte or word contents of the addressed memory location will be displayed in the data field and a decimal point (a dot) appears at the right edge of the data field indicating that the data can be updated. Note that when using the Examine Word command, the byte contents of the displayed memory location appear in the two least-significant digits of the data field and the byte contents of the next consecutive memory location (i.e entered memory address +1) appear in the two most significant digits of the data field.

6. If the contents of the memory location addressed are only to be examined, press the "." key to terminate the command, or press the "," (NEXT) key to examine the next consecutive memory location (Examine Byte Command) or the next two consecutive memory locations (Examine Word Command) or press the "PREV" key to examine previous byte or word location.

7. To modify the contents of an addressed memory location, enter the new data from the hexadecimal keyboard (entering the data values is discussed in detail in section 3.2).

8. The data displayed is not updated in memory until either the "," or "." key is pressed. If the "," key is pressed, the offset address and data contents of the next consecutive memory byte location (Examine Byte Command) or word location (Examine Word Command) are displayed. If the "." key is pressed, the command is terminated and the command prompt character is displayed in the address field indicating that the monitor is ready to accept a new command from the user.

Error Conditions:

Attempting to modify a non-existent or read-only (e.g. a PROM) memory location. Note that error is not detected until the "," or "." key is pressed. When an error is detected, the characters "Err" are displayed with the command prompt character, in the address field.

EXAMPLES

Example 1:- Examining a series of Memory Byte Locations starting from the location 10:50 (segment,offset).

Key	Address field	Data field	Comment
Reset	-ESA	86	System Reset
0 EB/AX	.		Examine byte command
1 ER/BX	1.		Segment address
0 EB/AX	10.		
:	10.		Segment/offset separator
5 0B/BP	5.		
0 EB/AX	50.		Offset value of the first memory location to be examined
,	50.	xx.	Memory data contents
,	51	xx.	Next memory location and data contents
,	52	xx.	Next memory location and data contents
PREV	51	xx.	Content of PREV location
.	--		Command terminator.

Example 2. Examining and modifying memory word location 1AAH relative to the CS-register. (Recall that if no segment value is specified, the default value is the current contents of the code segment register-CS)

Key	Address field	Data field	Comment
RESET	-ESA	86	System Reset
7 EW/DI	.		Examine Word Command
1 ER/BX	1.		
A	1A.		

/SS			
A /SS	1AA.		
,	1AA	x x x x.	Memory data contents
F	1AA	0 0 0 F.	
F	1AA	0 0 F F.	
A /SS	1AA	0 F F A.	
A /SS	1AA	F F A A.	
.	-		Data updated, command termination / prompt

To check that the data was updated successfully, press the EW key and enter the address 1AA. Press the "," key and note that "FFAA" is displayed in the data field.

Example 3: Examining and Modifying Memory Byte location 200H relative to the DS register.

Key	**Address field**	**Data field**	**Comment**
Reset 0	-ESA	86	System Reset
EB/AX	.		Examine Byte Command
REG	.		Register Input
9			
OW/DS	2000.		DS register
:	2000.		Segment/offset separator
2			
GO/CX 0	2.		
EB/AX 0	20.		Offset address
EB/AX	200.		
, C	200	x x.	Memory data contents
/IP C	200	0 C.	
/IP	200	C C.	New data to be entered
.	-		Data updated, command termination prompt

To check that data was updated successfully press the EB key and enter the address (DS:200H). Press the "," key and note that "CC" is displayed in the data field.

Example 4: Attempting to Modify PROM contents

Key	Address field	Data field	Comment
Reset	-ESA	86	System Reset
0			
EB/AX	.		Examine byte command
F F.			
F	FF.		Segment address
0			
EB/AX	FF0.		
0			
EB/AX	FF00.		
:	FF00.		Segment/offset seperator
1			
ER/BX	1.		Offset address
0			
EB/AX	10.		
,	10	xx	Data contents of the memory location
C			
/IP		C.	
B			
/ES		CB.	New data to be entered
,	-Err	.	Error message

Trying to modify the contents of a read-only memory location produced the error message. Repeat the above keying sequence to see that the memory contents of the location (FF00:10) remain unaltered.

3.5 EXAMINE REGISTER COMMAND

Function

The Examine Register (ER) command is used to examine and optionally modify the contents of any of the 8086/8088's registers.

Format

ER <reg key> [[<data>] ,]* [.]

Operation

1. To examine the contents of a register, press the ER key when prompted for a command entry.

2. Now a dot (decimal point) appears at the right edge of the address field. However, unlike in the EB or EW command, this prompt now means that a register name entry is required. Thus the subsequent hexadecimal keyboard entry will be interpreted as the register name (the acronym to the right of the slash sign on the key face) rather than as its hexadecimal value.

3. When the hexadecimal key is pressed, the corresponding register abbreviation will be displayed in the address field, the 16-bit contents of that register will be displayed in the data field and a data update prompt (dot) appears at the right edge of the data field.

Table 3.5 defines the 8086/8088 register names, the hexadecimal key board acronyms, the abbreviations appearing in the address field of the display and the sequence in which the registers are examined.

Table 3.5 8086/8088 Registers, acronyms and display abbreviations

Register Name	Keyboard Acronym	Display Abbreviation
Accumulator	AX	A
Base	BX	b
Count	CX	C
Data	DX	d
Stack Pointer	SP	SP
Base Pointer	BP	bP
Source Index	SI	SI
Destination Index	DI	dI
Code Segment	CS	CS
Data Segment	DS	dS
Stack Segment	SS	SS
Extra Segment	ES	ES
Instruction Pointer	IP	IP
Flag	FL	FL

4. When the register contents are displayed (with a dot at the right edge of the data field), the contents of this register can be modified if desired. To do this, enter the new value from the keyboard. This new value will be displayed in the data field and the register contents will be updated when either the`,' or `.' Key is pressed

5. If the "." key is pressed, the command is terminated and the command prompt character (-) is displayed. If the "," key is pressed, the abbreviation and contents of the "next" (according to the order of Table 3.5) register are displayed and opened for optional modification. Note that the sequence is not circular and thus pressing the "," key when the flag (FL) register is displayed will terminate the command.

EXAMPLES

Example1: Examining and Modifying a Register

Key	**Address field**	**Data field**	**Comment**
Reset	-ESA	86	System Reset
1			
ER/BX	.		Examine Register command
9			
OW/DS	dS	0 0 0 0.	Data segment register contents
1			
ER/BX	dS	0 0 0 1.	
0			

EB/AX	dS	0 0 1 0.	
.	-		Register updated, command termination prompt

Example 2: Examining a series of Registers.

Key	Address field	Data field	Comment
Reset	-ESA	86	System Reset
1			
ER/BX	.		Examine Register command
B			
/ES	ES	0 0 0 0.	Extra segment Register contents
,	IP	0 0 0 0.	Instruction pointer register contents
,	FL	0 0 0 0.	Flag Register contents
,	-		Command Termination / prompt

3.6 INPUT/OUTPUT COMMANDS

There are 4 commands available for Input/Output of Byte/Word data from/to a specified port. In entering the port address (in any of these four commands), it should be noted that 8086/8088 I/O addressing is limited to 64K (maximum address is FFFFH). Thus no segment value is permitted with the port address.

3.6.1 INPUT BYTE AND INPUT WORD COMMANDS

Function

The Input Byte (IB) and Input Word (IW) commands are used to input (accept) an 8-bit byte or 16-bit word from an input port.

Format

IB <port address> , [,]* .
IW <port address> , [,]* .

Operation

1. To use the Input Byte (IB) or the Input Word (IW) command, press the corresponding key when prompted for command entry.

2. When either key (IB or IW) is pressed, the decimal point appears at the right edge of the address field indicating that a port address entry is required.

3. Using the hexadecimal keyboard, enter the address of the port to be read. As already noted, no segment value is permitted with the port address.

4. After entering the port address, press the "," key. The input byte or word at the addressed port will be displayed in the data field.

5. Pressing the "," key again, updates the data field display with the current data byte or word at the addressed input port. Pressing the "." key terminates the command and the command entry prompt dash (-) appears.

Note:

The I/O ports provided on the ESA 86/88-2 and their addresses & usage are described in detail in chapter 5 on hardware.

EXAMPLE 1 : A Byte Input from port FFE0H

Key	Address field	Data field	Comment
RESET	-ESA	86	System Reset
4			
IB/SP		.	Input Byte Command
F	F.		
F	FF.		Port address
E	FFE.		
0	FFE0.		
,	FFE0	xx	Input Data Byte
.	-		Command termination prompt.

Example 2. Word Input from port FFE0

Key	Address field	Data field	Comment
RESET	-ESA	86	System Reset
8			
IW/CS	.		Input word command
F	F.		
F	FF.		
E	FFE.		Port address.
0	FFE0.		
,	FFE0	xxxx	Input Data Word
.	-		Command termination prompt

3.6.2 OUTPUT BYTE AND OUTPUT WORD COMMANDS

FUNCTION

The output Byte (OB) and output Word (OW) commands are used to output a byte or word to an output port.

Format

OB <port address> , <data> [, <data>] * .
OW <port address> , <data> [, <data>] * .

Operation

1. To use either command, press the corresponding hexadecimal key when prompted for command entry.

2. When either key (OB or OW) is pressed, a decimal point appears at the right edge of the address field indicating that a port address entry is required.

3. Using the hexadecimal keyboard, enter the desired port address. As mentioned already, no segment value is permitted with the port address.

4. After entering the port address, press the "," key. Now a decimal point appears at the right edge of the data field. This indicates that the data byte (OB command) or data word (OW command) to be output can now be entered.

5. Using the hexadecimal key board, enter the data byte or word to be output.

6. After entering the data, press the "." key to output the byte or word to the port and to terminate the command or press the "," key if additional data is to be output to the addressed port.

Notes:

As mentioned in the previous section, the I/O ports provided on ESA 86/88-2, their addresses and usage are explained in detail, in chapter 5, on Hardware.

Examples

Example 1: Output a byte to the port FFFDH

Key	**Address field**	**Data field**	**Comment**
RESET	-ESA	86	System Reset
5			
OB/BP	.		Output Byte Command
F	F.		
F	FF.		
F	FFF.		
D	FFFD.		Port Address
/FL			
,	FFFD .		
3	FFFD	03.	data byte to be output
ST/DX			
6	FFFD	36.	
MV/SI			
.	-		Command termination/prompt.

Example 2: Outputting 2 bytes of (count) data to Timer 0 of 8253. The port address is FFF9.

Key	Address field	Data field	Comment
RESET	-ESA	86	System Reset
5			
OB/BP	.		Output byte command
F	F.		
F	FF.		
F	FFF.		
9			
OW/DS	FFF9.		Output port address
,	FFF9	.	Data entry prompt
A			
/SS	FFF9	0A.	Output first byte
,	FFF9	.	
0			
EB/AX	FFF9	00.	Output second byte
.	-		command termination / prompt.

3.7 STEP COMMAND

FUNCTION

This command is used for single step execution of a program. In other words, this step (ST) command permits program instructions in memory to be executed individually. With each instruction executed, control is returned to the monitor from the program being executed.

Format

ST [<start address>] , [[<start address> ,* .

Operation

1. To use the step command, press the ST key when prompted for a command entry.

2. Now a starting address is displayed in the address field and an address entry prompt (dot) appears at the right edge of the address field.

3. If a starting address other than the address displayed is required, enter the desired address from the keyboard.

4. When the "," key is pressed, the instruction addressed is executed and the offset address of the next instruction to be executed is displayed in the address field and its associated instruction byte is displayed in the data field.

5. Again pressing the `,' key executes the current instruction and steps the program to the next instruction to be executed.

6. After executing an instruction, press the "." key to terminate the command. Control now returns to the monitor and the command entry prompt (-) appears.

Restrictions

1. An instruction that is part of a sequence of instructions that switches between stack segments (i.e. changes the SS and SP register contents) cannot be single stepped.

2. If an interrupt occurs prior to the completion of single-stepped instruction or if a single-stepped instruction generates an interrupt; when the monitor is re-entered, the CS and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (breakpoint) interrupt instruction (CCH) should not be single-stepped since its execution would step into the monitor.

To study this command, Single Step the programming examples presented in chapter 8 and observe the responses.

3.8 GO COMMAND

Function

The GO command is used to transfer control of the 8086/8088 from the keyboard monitor program to user's program.

Format

GO [<address>] [, <breakpoint address>] .

Operation

1. To use this command, press the GO key when prompted for command entry.

2. When the key is pressed, the current contents of the instruction pointer (IP) register are displayed in the address field, the byte contents of the memory location addressed by the IP register are displayed in the data field. Additionally, the decimal point appears at the right edge of the address field indicating that an alternate start address value can be entered.

3. If an alternate starting address is required, enter the address from the keyboard. (Note that in such a case, the data field is blanked).

4. To begin program execution press the "." key.

5. Now the monitor displays an "E" in the left-most digit of the address field and transfers control to the program (at the current instruction or at the alternate start address, if entered by the user).

6. To exit from the executing program and return control to the monitor, you can press either the system RESET key or INTR key. If the RESET key is pressed, control returns to the monitor and the appropriate 8086/8088 registers are initialized (i.e the earlier values are lost). However, if the INTR key is pressed, monitor gets the control, all of the 8086/8088 registers are saved and the monitor prompts for a command entry. As all the register values are saved, by pressing the GO key now, you can resume the execution of the program from the point at which you halted it (by pressing the INTR key).

   If the user program has an INT 3 instruction (opcode CCH), and it is executed, control returns to the monitor and all the registers are saved. Usually, this is the method by which user programs return control to monitor. (Ref. chapter 8 on programming examples.)

Breakpoint facility:

The GO command permits the optional specification of a "breakpoint address". The effect is as if the KB INTR key is pressed when the program reaches the breakpoint address. In other words, when program reaches the breakpoint address, control goes to the monitor, all registers are saved and the monitor prompts for command entry.

To use this facility, press the NEXT key after entering the starting address and then enter the breakpoint address. No segment value is permitted with breakpoint address. Thus the default segment value is either the starting address segment value (if specified) or the current contents of the CS register (if no segment value is specified with the starting address). After entering the break point address press the "." key.

The break point facility is implemented as follows. After the "." is pressed, the monitor replaces the instruction at the breakpoint address with an interrupt instruction and saves the "breakpoint" instruction before transferring control to the user's program. When the program reaches the break point address, control is returned to the monitor. The monitor saves all the registers, restores the breakpointed instruction and prompts for command entry. Consequently -

1) You cannot breakpoint an instruction in the read-only memory and
2) Breakpoint address must be specified each time the program is to be executed with a breakpoint.

ERROR CONDITIONS

Attempting to breakpoint an instruction in read-only memory. To study this command, execute the examples in chapter 8,with breakpoints and observe the response.

3.9 MOVE COMMAND

Function

This command (MV) can be used to move a block of data from one portion of the memory to another portion of the memory.

Format

MV <Start address> NEXT <end address> NEXT <destination address>.

Operation

1. To use the move command press the MV key when prompted for command entry.

2. When the key is pressed, three dots appear in the address field indicating that three address entries are required. As can be seen from the format of this command, the three addresses to be entered are, in order:
   a) The starting address of the block of data to be moved.

   b) The ending address of the block of data to be moved.

   c) The destination address i.e. the starting address of the portion of the memory into which the block of data is to be moved.

3. As noted above, upon pressing MV key, three decimal points appear in the address field. Each time an address value is entered the leftmost decimal point disappears. Thus the number of decimal points lit any one time indicates the number of address entries still required.

4. Enter the three address values as required and then press the "." key.

5. Now the data is moved and the command prompt sign (-) is displayed.

NOTES: There are many points worth noting about this command.

1. Though any value can be specified for the start address, no segment value is permitted with an ending address. Consequently block move is limited to 64k bytes

2. When the block of data is moved, the data contained in the source block is moved (simply copied) one byte at a time into the destination block. Thus:

   a) The data contained in the original (source) memory locations is not altered unless the destination address falls within the original block of data.

   b) If the destination address falls within the original block of data, then the overlapping memory locations will be overwritten by the data moved.

   c) From the above discussion, it is clear that the Move command can be used to fill a block of memory with a specified constant. This can be achieved by specifying a destination address that is one greater than the start address. Then, the block of memory locations from start address to end address +1 will be filled with the value contained in the start address location.

(The Examine Byte Command can be used to specify the desired constant as the contents of the start address)

ERROR CONDITIONS

Attempting to move data into Read Only or non-existent memory.

EXAMPLES

Example 1: In chapter 8, Section 8.3, there is an example program (Example 6 CORRECT GUESS), assembled from locations 2000 to 20AF. Move it to locations starting at 2200H.

Key	Address field	Data field	Comment
RESET	- E S A	8 6	System Reset
6			
MV/SI	. . .		Move command
2			
GO/CX	. . 2.		
0			
EB/AX	.2. 0.		Start address (2000H)
0			
EB/AX	2.0. 0.		
0			
EB/AX	20.0. 0.		
NEXT	. .		

2			
GO/CX	.2.		End address (20AFH)
0	2.0.		
EB/AX			
A	20.A.		
/SS			
F	20A.F.		
NEXT	.		
2			
GO/CX	2.		Destination (2200H)
2			
GO/CX	22.		
0	220.		
EB/AX			
0	2200.		
EB/AX			
.	--		Program moved command prompt.

Example 2: Fill the memory locations 2000H to 20FFH with the constant 55H

Key	Address field	Data field	Comment
RESET	- E S A	8 6	System Reset
0	.		Examine Byte Command
EB/AX			
2			
GO/CX	2.		First location
0			
EB/AX	2 0.		
0			
EB/AX	2 0 0.		
0			
EB/AX	2 0 0 0.		
NEXT	2 0 0 0.	X X.	Contents of Location 2000
5			
OB/BP	2 0 0 0.	5.	
5			
OB/BP	2 0 0 0.	5 5.	New value (55H)
.	--		Command prompt
6			
MV/SI	. . .		Move Command
2			
GO/CX	. . 2.		Start address = 2000H
0			
EB/AX	.2.0.		
0			
EB/AX	2.0.0.		
0			
EB/AX	2.0.0.0.		
NEXT	. .		
2			
GO/CX	.2.		End Address = 20FEH
0			
EB/AX	2.0.		

F	20.F.	
E	20F.E.	
NEXT	.	
2		
GO/CX	2.	Destination = 2000+1= 2001H
0		
EB/AX	20.	
0		
EB/AX	200.	
1	2001.	
ER/BX		
.	--	2000 to 20FFH = 55H

3.10 USE OF REG , + AND - KEYS.

The hex keypad includes "REG","+" and "-" keys which can be used in a variety of combinations to simplify the address/data entry. This section illustrates some of the typical uses of these keys. Once familiar, user can find many other situations where these keys can be effectively used.

3.10.1 USE OF "+" KEY.

This key allows the user to add two hexadecimal values. This function, thus simplifies relative addressing by allowing the user to readily calculate an address location relative to a base address. Further, during the data entry, arithmetic expression results can be easily stored.

To illustrate the use of this key, consider the programming example 1 of section 8.2. Here at address 200EH, the instruction is as follows:

Location	Contents	Instruction
200E	E2,F8	LOOP AW

Suppose, we want to examine the instruction at the target address of Loop instruction. One way is to calculate the address manually as 10H + F8H = 08H and examine the contents of 2008H. However, we can directly specify the address using the following key sequence.

EB 2 0 1 0 + F F F 8 NEXT

Thus the address is directly specified as 2010H + FFF8H (Note that short Jump specifies only low byte increment i.e F8H; it must be sign-extended to FFF8H !) , instead of manually calculating the value.

Generally, relative-addresses can thus be entered directly, without calculating the actual values. More examples illustrating the use of `+' key are given later.

3.10.2 USE OF "-" KEY

The use of this key is similar to the use of the '+' key. It allows the subtraction of one hexadecimal value from another, thus simplifying relative address calculations.
As an illustration, consider the same Example 1 from Section 8.2. Now suppose, you are entering the code for instruction LOOP AW at address 200EH. The opcode for LOOP instruction is E2H but what should be the relaltive address value ? It can be computed as 08H (Low byte of address AW) minus 10H (LOW byte of instruction following LOOP AW instruction). But one can avoid the calculation of the relative address by directly specifying the arithmetic operation, using the following key sequence.

E B 2 0 0 F NEXT 0 8 - 1 0 NEXT .....

The system calculates 08H -10H as F8H and stores this value in the location 200F H!

3.10.3 USE OF "REG" KEY

This key allows the user to specify the contents of any of the 8086/8088's registers in the calculation of an address or data value.

To illustrate, consider the Example 4 of section 8.2. Though this example was presented there assuming serial mode of operation, the same code can be executed in keyboard mode also. Assume that we are running the program in keyboard mode. This program searches through a byte array for a specified value and returns the result (00H - not found, FFH - found) in the register AL. Further if the value is found, BX contains the offset in the array of the matching element.

Suppose after executing the above program, you want to examine the element in the array. One method would be to find out the offset by examining the contents of BX and find out the segment value by examining the contents of ES. Suppose (ES)= 0220H and (BX) = 0004H after the execution of the program. Then we can examine the matching element by the following key sequence.

EB 220 : 4 NEXT

However, use of the 'REG' key simplifies the whole operation. We want to examine the data at (ES) : (BX) . Specify it directly ! use the following key sequence.

EB REG ES : REG BX NEXT

As can be seen from the above discussion, REG key simplifies the specification of an address relative to a register by allowing the direct entry of the register name.

Further, 'REG', '+' and '-' keys can be combined in several different ways to simplify the address/data entry.

As an illustration suppose, you want to move 128(decimal) bytes starting from 0:2100 to a block starting at segment value equal to ES register plus 10H and offset value of 40H. One way is to calculate all addresses as follows.

Starting address of source=0 : 2100 Ending address of source =2100H + 7FH = 217FH
Assume (ES) = 0270H
Starting address of destination = ((ES) + 10H) =) 0280H : 40

Now use move command by the following key sequence
MV 0 : 2100 NEXT 217F NEXT 280 : 40 EXEC

However, the complete procedure can be simplified by specifying the address directly as shown by the following key sequence
MV 0 : 2100 NEXT 2 1 0 0 + 7 F NEXT REG ES + 10 : 40 EXEC

As already mentioned, the above examples are for the purpose of illustration only. With experience, the user can find a variety of situations where proper use of 'REG', '+' and '-' keys can greatly simplify the entry of address/data values.

3.11 USER PROGRAMMABLE FUNCTION KEYS

ESA 86/88-2 provides 7 user programmable function keys F1 to F7. When a User programmable function key is pressed monitor transfers control to a program whose address is assumed to be available in certain pre-specified RAM locations as shown in Table 3.6. NOTE that these addresses are pointers in the form CS:IP.

Following power ON/Reset the monitor initializes these locations to the starting addresses of routines given by Table 3.7. The user can make use of these keys to control transfer to his program by making changes in the locations given in Table 3.6. Note that F7 is always a user- programmable key.

NOTE: Power ON or pressing RESET key will reinitialize these keys to the default values.

TABLE 3.6

KEY	ADDRESS Location holding IP value		Location holding CS value	
F1	190	191	192	193
F2	194	195	196	197
F3	198	199	19A	19B
F4	19C	19D	19E	19F
F5	1A4	1A5	1A6	1A7
F6	1A8	1A9	1AA	1AB
F7	1AC	1AD	1AE	1AF

TABLE 3.7

FUNCTION KEY	ROUTINE	COMMENT
F1*	BLANK CHECK	OPTIONAL EPROM
F2*	READ	PROGRAMMER
F3*	PROGRAM	COMMANDS
F4*	VERIFY	(Refer Chapter 10 onPROM PROGRAMMER SYSTEM)
F5*	TAPE READ	OPTIONAL AUDIO CASSETTE INTERFACE COMMANDS
F6*	TAPE WRITE	(Refer chapter 9 on Audio Tape interface)
F7	-Err	User-programmable function key

* Optional (for proper operation, the optional hardware should be installed)

CHAPTER 4

SERIAL MONITOR

4.1 INTRODUCTION

This chapter describes the commands supported by the Serial Monitor program. It is designed to operate with a CRT Terminal/Host Computer that has been connected to the serial interface RS 232 C connector J3 (refer to chapter 5 on Hardware, for details).

The system, on power-ON goes to serial mode of operation if the switch position 8 of the on-board DIP switch is OFF or the system can be changed from keyboard mode to serial mode of operation any time by putting the same switch OFF and pressing the RESET key (Refer chapter 2 for further details).

The serial transmission baud rate is set through on - board DIP switches. (The details of its setting for different baud-rates are given in Chapter 2, Installation and configuration).

4.2 GENERAL OPERATION

Following power-ON or whenever the RESET key is pressed, the monitor jumps to an initialization routine.

The routine initializes the 8086/8088's CS(Code Segment), DS (Data Segment), ES (Extra Segments) SS (Stack Segment), IP (Instruction pointer) and FL (Flags) registers to 0H and the SP (Stack Pointer) register to 100H.

Whenever the RESET key is pressed, all information about the user program is lost. However the contents of the user portion of the RAM are not disturbed.

Further, interrupt vectors 1,2 and 3 are initialized as follows:

Interrupt 1: Single Step: used with the single step command

Interrupt 2: NMI (Non-Maskable Interrupt): connected to Key board KB INTR key.

Interrupt 3: Breakpoint: used with the GO command.

A detailed discussion of the 8086/8088 interrupt vectors can be found in 8086 Family User's Manual (chapter 2 Processor Control and Monitoring. Interrupts, PP 2.22 - 2.28)

Whenever the monitor is re-entered as a result of Single Step, NMI or Breakpoint Interrupt, the monitor saves the contents of the 8086/8088 registers on the user stack and subsequently restores the register contents from the stack before it prompts for command entry. Since the SP register is initialized to 100H, and memory locations 0H-CFH are reserved for monitor, the stack length reserved for the user is 48 bytes (locations D0H -FFH). Of these, 26 bytes must be left for register contents should one of the above interrupts occur (leaving 22 bytes for the user). In addition, locations 100H-1FFFH are used by the system to store certain variable parameters. Hence the user area of RAM starts from 0:2000H.

When the system enters Serial Mode of operation, the sign-on message **'ESA-86 MONITOR Vx.y'** (if 8086 is installed; if 8088 is installed, the sign-on message would be ESA-88 MONITOR Vx.y), is displayed (x is current version number and y is the revision number) on one line and a period ('. ') on the

next line indicating that the monitor is ready to accept commands from the user. With the exception of RESET and INTR keys, the keyboard is disabled.

On board eight seven segment display would show "E SE 86" or "E SE 88" depending on the CPU installed.

4.3 STRUCTURE OF MONITOR COMMANDS

When the monitor is ready to accept a command from the user it outputs a period ('.') as the command prompt character at the beginning of a new line.

The command entered by the user consists of either a one or two-character command mnemonic followed by a list of command parameters. This list may consist of one to three parameters depending on the particular command being used. When more than one parameter is required, a single comma (',') is used between the parameters as a separator. A command is terminated either by a carriage return or by a comma, depending on the command itself. Commands are executed one at a time and only one command is allowed in a command line.

4.3.1 PARAMETER ENTRY

When a command requires the entry of an address/data value from the user, the following rules apply.

All addresses in the 8086/8088 system consist of a segment value and an offset value. The segment value is entered first, a colon (':') is entered as a seperator and then the offset value is entered. If the segment value is not specified (note that in such a case, the colon also should be omitted), the default segment value is the current contents of the code segment (CS) register. Enter the address as hexadecimal value , most significant character first. The valid range of hexadecimal values for an address entry (either segment or offset) is from 0000 to FFFFH. If more than four digits are entered, only the last four digits entered are valid. In other words all address values are interpreted modulo 64K.

Data is also entered as hexadecimal value, most significant character first. The valid range for data entries is 00 to FFH for byte entries and from 0000 to FFFFH for word entries. If more than two (for byte entries) or four (for word entries) digits are entered, only the last two or four digits entered are valid.

All the commands except the X (Examine / modify register) command require only hexadecimal values as parameters. The register abbreviation entries required by the X command are described later while describing the X command in detail.

4.3.2 USE OF +,- OPERATORS AND REGISTER, IN SPECIFICATION OF PARAMETERS

While the address/data parameters are to be entered as hexadecimal values, it is possible to use the operators '+' and '-' to form expressions and also to specify the use of the contents of any 8086/8088 register as address/data values.

For example, suppose we wish to specify an address whose segment is (ES) + 10H and whose offset is (BX) - 30H.One way is to calculate these values as shown below.

Assume (ES)	= 0270H and (BX)	= 0080H
Segment value	= (ES)+ 10H	= 0280H
Offset value	= (BX) - 30H	= 0050H

Then the address can be specified as 0280:0050

However, ESA 86/88-2 serial monitor supports the direct specification of such an address value as shown below:

ES + 10: BX - 30

The system will automatically evaluate such expressions and use the correct values. These operators can be used to considerably simplify the specification and checking of relative addresses also. These uses are described in greater detail in section 3.10.

4.3.3 RESPONSE TO ERRORS

Whenever an error is detected the command is aborted, the symbol (`?') is output on the command line, a carriage return and linefeed are issued and the command prompt character (`.') is output at the beginning of the new line.

Command execution occurs only after a command terminator (a comma or a carriage return depending on the command) is entered. Hence a command entry can be cancelled any time before the terminator is entered by "committing an error". That is, enter any character that is not legal for the entry expected. The monitor detects this error, aborts the command, displays `?' symbol and returns to command entry mode.

4.4 MONITOR COMMANDS

The Serial Monitor can execute 17 individual commands. These commands are summarized in Table 4.1 and are described in detail in the sections which follow.

In both the table and individual command descriptions, the following notation is used.

[V] indicates that`V' is optional
[V]* indicates one or more optional occurences of "V"
<V> indicates that "V" is a parameter to be entered by the user.
,/- indicates that one of the two characters is to be entered.

These symbols are used to clarify the command formats and they are to be neither entered by the user nor output by the system.

In the description of the individual commands, it is assumed that 8086 CPU has been installed. However, all the commands work in identical fashion even if 8088 CPU is installed. The only observable difference would be in the sign-on message produced by the system following power ON/Reset.

TABLE 4.1
SUMMARY OF SERIAL MONITOR COMMANDS

COMMAND	FUNCTION/FORMAT
S(substitute Memory bytes)	Displays / modifies memory byte locations **S <address> <cr>, [[<new data>] ,/-] * <cr>**
SW(substitute Memory Words)	Displays / modifies memory word locations **SW <address> <cr> , [[<new data>] ,/- <cr>**

D(Display Memory bytes)	Displays block of memory in byte format **D <start address> [,<end address>]<cr>**
DW(Display Memory Words)	Displays block of memory in word format **DW <start address> [,<end address>] <cr>**
X(Examine / modify Register)	Displays modifies 8086 registers **X [<reg>]<cr> [[<new data>],] <cr>**
M (Move)	Moves a block of memory contents **M <start address> , <end address> ,** **<destination address> <cr>**
I (Input byte)	Accepts and displays the data byte at the input port **I <port address> <cr> , [,] * <cr>**
IW(Input Word)	Accepts and displays the data word at the input port **IW <port address> <cr> , [,] * <cr>**
O (output byte)	Outputs the data byte to the output port **O <port address> <cr>, <data> [,<data>] * <cr>**
OW (Output Word)	Outputs data word to output port **OW <port address> <cr> , <data> [,<data>]* <cr>**
G (Go)	Transfers the processor control from the monitor to user program with optional break point. **G <cr> <start address> [, <breakpoint address>] <cr>**
N (Single Step)	Executes users program single instruction **N <cr> [<start address>] , [[<start address>],] <cr>**
R (Read hex file)	**R [<bias number>] <cr>**
W (Write hex file)	**W[X]<start address>,<end address> [,<exec address>]<cr>**
T (Save onto Audio Cassette interface)*	**T <cr> <file name>,<start address>,<end address> <cr>**
L (Load from Audio Cassette interface)*	**L <cr> <file name> <cr>**
P (Enter PROM Programmer)**	**P <cr>**
A (Enter Assembler)***	**A <cr>**

** **Refer Chapter 9***
*** **Refer Chapter 10***
**** **Refer Chapter 7***

4.4.1 SUBSTITUTE MEMORY COMMANDS

FUNCTION
The substitute memory byte (S) and substitute memory word (SW) commands are used to examine the contents of the selected memory locations. If the contents of the memory location can be modified (e.g, a RAM location), the contents can optionally be modified.

FORMAT

S <address> <cr> *, [[<new data>],/-]* <cr>*
SW <address> <cr> *, [[<new data>],/-]* <cr>*

OPERATION

1. Both the commands operate in a similar fashion. To use either command, enter S or SW when prompted for command entry. Then enter the address of the memory location to be examined. After the address value is entered, enter a comma and a carriage return.

2. The monitor will now output the contents of the addressed location followed by a dash (the monitor's data entry prompt character) and a space to indicate that the addressed location can now be modified.

3. To modify the contents of this location, enter the new data value. Note that when using the SW command, the byte contents of the next consecutive memory location (addressed memory location +1) are output first, followed by the byte contents of the actual location addressed. Similarly, when updating memory contents using the SW command, the first byte entry will be written into the next consecutive memory location, and the second byte entry will be written into the addressed memory location.

4. After optionally modifying the contents of the addressed location, enter a comma or minus (-) to examine / modify the next consecutive memory location or previous location (S command) or next two consecutive memory locations or previous two memory locations (SW command)

5. A carriage return at any stage terminates the command.

ERROR CONDITIONS:

1. Trying to modify non-existent or ROM locations.

EXAMPLES:

1. Examine RAM location 2100 H, relative to the DS register, modify the contents of location 2101 H and 2102 H and examine the contents of 2101H again.

```
.S DS:2100, <cr> A5-,
2101  FF-B7,
2102  FF-SA-
2101  B7-<cr>
```

2. Examine ROM location FF00:9CH.
   .S FF00:9C, <cr> FA- <CR>
   .
3. Trying to modify ROM location
   .S FF00:9C, <cr> FA-FB, ?
   .
4. Examine word at location 2120 H relative to DS register
   .SW DS:2120, <cr> A1F4- <cr>
   .

(Note that here A1H is at location DS:2121 H while the byte at the addressed location DS:2120 is F4H)

4.4.2 DISPLAY MEMORY COMMANDS

FUNCTION

The Display memory byte (D) and Display memory word (DW) commands are used to output the contents of a block of memory, either in byte or word format.

FORMAT

D <start address> [,<end address>] <cr>
DW <start address> [,<end address>] <cr>

OPERATION

1. Both the commands operate in a similar fashion. These commands provide a line-formatted output of the memory block starting at the `start address' and ending at the 'end address'.

Note that the `end address' is always relative to the segment value specified with "start address", or implied with `start address' (i.e the contents of CS register if no segments value is specified). And thus with each command execution, a maximum of 64K bytes or 32K words can be displayed.

To use either command, enter D (for byte output) or DW (for word output) when prompted for command entry. Then enter the start address of the memory block. Now, if only one byte or word is to be displayed, enter the carriage return. On the other hand, if a block of memory is to be displayed enter the end address and then the carriage return.

2. The monitor will now output, beginning on the next line, the starting offset address, the data contents of that location, and, if end address is specified, the contents of the consecutive memory locations upto (and including) the end address.

   The line format is arranged in such a way that any subsequent lines (if present) will begin with an offset address whose last nibble is zero. A line consists of a maximum number of 16 bytes or 8 words.

3. The display memory commands can be cancelled, or the output can be stopped and resumed at any time by entering appropriate control characters from the console.

Control-C, entered at any time, immediately terminates the command. Monitor returns to the command entry mode.

Control-S stops the output but does not terminate the command. Entering **Control-Q** now resumes the output from the point at which it has been stopped. Now also, you can enter **Control-C** to terminate the command.

Note that, after entering control-S, you can follow it only with control-Q or control-C. No other console input is allowed following a control-S.

ERROR CONDITIONS:

1. Specifying an end address that is less than the offset value of the start address.

EXAMPLES:

1. Display contents of location 140H relative to DS register

```
.D DS:140 <cr>
0140 0F
.
```

2. Display contents of locations 10CH through 125H relative to the DS register.

```
.D DS:10C, 125 <cr>

010C F0 00 00 9C
0110 EA BD 04 BA 00 F2 10 21 A4 35 14 17 FA F0 F1 F5
0120 30 31 32 32 34 35
.
```

3. Display word at location 10C H relative to DS register

```
.DW DS:10C <cr>
010C 00F0
```

4.4.3 EXAMINE/MODIFY REGISTER COMMAND

FUNCTION

The Examine/Modify Register (X) command is used to examine and optionally modify the contents of any of the 8086/8088's registers.

FORMAT

X [<reg>] <cr> [[<new contents>] ,]* <cr>

OPERATION

1. To use the Examine/Modify Register command, enter X when prompted for command entry.

2. If you wish to examine the current contents of all the registers, enter a carriage return. Now the monitor will output the contents of all 14 registers.

3. If you wish to examine/modify the contents of a particular register, enter the abbreviated register name after entering X and press <cr>. (The abbreviations for the register names are shown in Table 4.2)

TABLE 4.2
Register Abbreviations

Register Name	Abbreviation
Accumulator	AX
Base	BX
Count	CX
Data	DX
Stack Pointer	SP
Base Pointer	BP
Source Index	SI
Destination Index	DI
Code Segment	CS
Data Segment	DS
Stack Segment	SS
Extra Segment	ES
Instruction Pointer	IP
Flag Register	FL

Now the monitor will output an equal sign ("="), the current contents of the specified register, the data prompt character ("-") and a space.

If you wish to change the contents of this register, enter the new contents followed by a comma, or a carriage return.

Entering a carriage return terminates the command.

Entering a comma displays the contents of the next “sequential” register and opens it for optional modification. The register sequence is in order as shown in Table 4.2 . Note that the sequence is not circular and that if a comma is entered after the contents of the "last" register (i. e. Flags register FL) is examined /modified, the monitor returns to the command entry mode.

EXAMPLES:

1. Examine the contents of all the registers.

```
.X <cr>

AX = F182   BX=A3C3   CX=000F   DX=1242   SP=0100   BP=4020   SI=6020
DI = 6F20   CS=0000   DS=0000   SS=0000   ES=0000   IP=0200   FL=F046
.
```

2. Examine and modify the SP register and examine the next register i.e BP

```
.XSP=0100-0110,
BP=4020-<cr>
.
```

4.4.4 MOVE COMMAND

FUNCTION

The Move command is used to move a block of data from one area of the memory to another area.

FORMAT

M<start address>, <end address>, <destination address> <cr>

OPERATION

1. To use the Move command, enter M when prompted for command entry.
   Now enter the three required parameters separated by commas. The three parameters are the "start" and "end" addresses of the memory block to be moved, and the start address of the destination block.
   Note that, as with the D (Display Memory) command, end address is relative to the segment value specified with `start address' or implied with the `start address' (i.e the contents of CS register if no segment value is specified). Consequently, no segment value is permitted with the `end address' and with each command execution, a maximum of 64K bytes can be moved.

After entering the three parameters enter the carriage return.

2. Now the monitor moves the contents of the memory block from "start address" to "end address", to consecutive memory locations beginning at `destination address'.

After moving the block of memory, monitor returns to the command entry mode.

Notes:

Since a move is performed one byte at a time, M command can be used to fill a block of memory with a predefined constant. To do this, use S command to move the constant into the "start address". Then use M command with a "destination address" which is one greater than the "start address". Then the memory block from start address to "end address" +1 is filled with the value contained in "start address".

ERROR CONDITIONS:

1. Trying to move data into a non-existent or read-only (e.g.,ROM or PROM) memory location.

2. Specifying an `end address' value which is less than the offset value of the "start address".

EXAMPLES:

1. Move the contents of the locations 2000H through 2020H relative to CS register, to the memory block starting at 3000H relative to the DS register.

   .M 2000, 2020, DS:3000 <cr>

2. Fill memory locations 2000H through 2100H relative to DS register, with constant AAH.

   .SDS:2000, BA-AA<cr>
   .MDS:2000,20FF,DS:2001<cr>
   .

4.4.5 INPUT BYTE AND INPUT WORD COMMANDS

FUNCTION

The Input Byte (I) and Input Word (IW)commands are used to input (accept) a byte or word from an input port, and to display the accepted byte or word.

FORMAT

I <port address> <cr> , [,]* <cr>
IW<port address> <cr> , [,]* <cr>

OPERATION

1. Both the commands operate in a similar way. Enter I (to input byte) or IW (to input word) when prompted for command entry.

Now enter the address of the port to be read. As already noted, I/O space is only 64K bytes and thus no segment value is permitted with 'port address'

2. After entering port address and <cr> , enter a comma. The Monitor reads the byte or word at the specified port and displays it on the console.

3. Each subsequent comma causes the current data at the addressed input port to be displayed on a new line.

A carriage return terminates the command and the monitor returns to the command entry mode.

NOTES

1. The I/O ports provided on ESA 86/88-2, their addresses and usage is summaraized in section 5.4.6.

Examples:

1. Input a byte from parallel I/O port at address FFE1H

```
.IFFE1 <cr> ,
FA<cr>
.
```

2. Input a series of words from parallel I/O ports located at addresses FFE0H AND FFE1H.

(Note that when using word input i.e IW command, low order address is entered as port address)

```
.IWFFE0 <cr> ,
A2A2,
B2B2,
C2C2<cr>
.
```

4.4.6 OUTPUT BYTE AND OUTPUT WORD COMMANDS

FUNCTION

The Output Byte (O) and Output Word (OW) commands are used to ouput a byte or word to an output port.

FORMAT

O <port address> <cr> , <data> [, <data>] * <CR>
OW<port address> <cr> , <data> [, <data>] * <CR>

OPERATION

1. Both the output commands (O and OW) operate in a similar way. When prompted for a command entry, enter O to output a byte, or OW to output a word.

   Now enter the address of the port to which data is to be output. Similar to input commands, no segment value is permitted with the port address. (I/O address space is limited to 64K bytes).

2. After entering the port address, enter a comma, followed by the data to be output.

3. Now enter a comma or carriage return.
   Entering a carriage return causes the data to be output to the specified port, terminates the command and returns monitor to the command entry mode. Entering a comma causes the data to be output to the specified port but does not terminate the command, and it permits subsequent data output to the specified port. Data can be output to the specified port by entering new data followed by a comma. At any stage a carriage return following a data entry outputs the data and terminates the command returning to the monitor command entry mode.

NOTES

As mentioned already, the I/O ports provided on ESA-86/88-2, their use and addresses are summarized in section 5.4.6.

Examples:

1. Program the parallel I/O port FFE0H for output by sending appropriate command byte to the status port FFE6H.

```
.OFFE6 <cr> , 80<cr>
.
```

2. Output a series of bytes to 8253 status/command port

```
.OFFFF,36,76,36,<cr>
.
```

4.4.7 GO COMMAND

FUNCTION

The Go (G) command is used to transfer control of the 8086/8088 from the Serial monitor to an user's program.

FORMAT

G <cr> [<start address>] [, <breakpoint address>] <CR>

OPERATION

1. To use the GO command, enter G when prompted for command entry.

2. When G is entered followed by a <cr> , the monitor outputs the current IP (instruction pointer) register contents, the data entry prompt character ("-") and the byte contents of the memory location addressed by the IP register.

3. If an alternate starting address is required, enter the "start address".

4. To begin program execution enter a carriage return. Control is now transferred from the monitor to the user program.

5. To exit from the executing program and to return control to the monitor press either the system reset (RESET) key or the KBINTR key.

   If the RESET key is pressed, control is transferred to the monitor program and the appropriate 8086/8088 registers are reinitialized.

   If the KBINTR is pressed, the program is interrupted, the serial monitor is re-entered, and all of the 8086 / 8088 registers are saved. The monitor now outputs the commercial at sign (`@'), followed by the address of the next program instruction to be executed when the KBINTR key was pressed (the displayed address consists of the current contents of the CS (code segment) register, a colon, and the current contents of the IP (instruction pointer) register). The address is followed by the display of all processor registers, a disassembled line of next program instruction followed by a command prompt "." You can resume the program again from this point at which you interrupted it, by entering the GO command followed by a carriage return.

4.4.7.1 BREAKPOINT FACILITY:

The GO command permits the optional specification of the "breakpoint address". The effect is as if the KBINTR key is pressed when the program reaches the breakpoint address. In otherwords, when program reaches the breakpoint address, control goes to the monitor and all registers are saved. Further the monitor outputs the message **br** followed by the commercial at sign (`@'), followed by the address of the breakpointed instruction (this address is displayed as the current contents of CS register, a colon and the current contents of IP register). This message is followed by the display of all processor registers, a disassembled line of next program instruction followed by command prompt.

If a subsequent GO command is entered followed by a carriage return, execution resumes at the break pointed instruction.

To use this facility, enter a comma after optionally specifying an alternate start address and then enter the breakpoint address followed by a carriage return. Note that no segment value is permitted with breakpoint address. Thus the default segment value for the breakpoint address is either the "start address" segment value (if specified) or the current CS register content (if a segment value is not specified with "start address" or if no alternate start address is given).

The breakpoint facility is implemented as follows: The monitor saves the instruction at the breakpoint address, and replaces it with an interrupt instruction before transferring control to the user's program. When the program reaches the breakpoint address, control is returned to the monitor. The monitor saves all registers, restores the breakpointed instruction and displays the message already described. It then issues the command entry prompt allowing the user to modify any of the registers, memory locations before resuming the execution of the program.

From the above description, it is clear that

1. You cannot breakpoint an instruction in the read-only memory and ,

2. Break point address must be specified each time a program to be breakpointed is executed.

ERROR CONDITIONS:

1. Attempting to breakpoint an instruction in read-only memory.

Examples:

1. Transfer control to the program at 2000H relative to the CS register

```
.G<cr>01EF-E9 2000<cr>
.
```

2. Transfer control to the program at location 20:300H and breakpoint the instruction in location 20:3F2H.

```
.G<cr>01EF-E9 20:300,3F2 <cr>

BR @ 0020:3F2
AX=F5BF BX=FFFF CX=FFFF DX=DFBF SP=0100 BP=4020 SI=6020
DI=6F20 CS=0020 DS=0000 SS=0000 ES=0000 IP=03F2 FL=F046
------ 03F2 90                    XCHGW  AX,AX;
```

3. Effect of pressing the KBINTR key

```
.G<cr>01EF-E9 20:300<cr>
```

Suppose, at some point while program is running, you pressed the KBINTR key, then the system displays

```
@  0020:322
AX=F5BF BX=FFFF CX=FFFF DX=DFBF SP=0100 BP=4020 SI=6020
DI=6F20 CS=0020 DS=0000 SS=0000 ES=0000 IP=0322 FL=F046
------ 0322 90                    XCHGW AX,AX;
```

4.4.8 SINGLE STEP COMMAND

FUNCTION

The Single step (N) command is used to execute a single instruction of a program. After each instruction executed, control is returned to the monitor from the program being executed.

FORMAT

N <cr> <start address>, [[<start address>],]* <cr>

OPERATION

1. To use the single step command, enter N with <cr> when prompted for command entry.
2. Now the monitor will output the current contents of instruction pointer (IP) register and the instruction byte pointed to by the IP (and CS) register.
3. If the execution of an instruction at an address other than the displayed address is required, enter 'start address' (if `start address' does not include a segment address, current contents of CS register are used; if `start address' includes a segment value, both CS and IP registers are modified).
4. Now enter a comma. The addressed instruction is executed and control returns to the monitor. The monitor saves all the registers, displays them and disassembles and outputs the next instruction. On the next line, it displays the address of the next instruction (IP register contents) and the byte at that address.
   Each time a comma is entered, the addressed instruction is executed and the address and instruction byte contents of the next instruction to be executed along with all registers and disassembly of next opcode are displayed.
   A new start address can be specified after executing any instruction.

5. A carriage return, after executing an instruction terminates the command and returns the monitor to command entry mode.

RESTRICTIONS

1. An instruction that is part of a sequence of instructions that switches between stack segments (i.e changes the SS and SP register contents) can not be single stepped.

2. If an interrupt occurs prior to the completion of a single-stepped instruction or if a single-stepped instruction generates an interrupt, when the monitor is re-entered, the CS and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (breakpoint) interrupt instruction (CCH) should not be single-stepped (since its execution would step into the monitor).

EXAMPLES

For examples to use this command, single step the Example programs given in Chapter 8 and observe the results.

4.4.9 READ HEX FILE COMMAND

FUNCTION

The Read Hex file (R) command allows the monitor to read hexadecimal object file from paper tape and to load the data read from the file into the memory.

FORMAT

R [<bias number>] <cr>

OPERATION

1. To use the Read Hex File command, enter R when prompted for command entry, when the tape is loaded in the reader and ready, enter a carriage return. The data read from the file will be written into memory beginning at each record's load address. If the file is in the 8086 format and includes an execution start address record, the CS and IP registers will be updated with the execution address specified in that record. If the file is in the 8080 format and includes an EOF (end of file) record, the IP register is updated with the execution address specified in the EOF record. Note that the a segment value is not used with the 8080 file format. The data read is written into memory locations relative to the segment value of zero and, when an EOF record execution address is specified, the CS register is not changed.

When an optional bias is specified, it is added to each record's load address.

ERROR CONDITIONS

1. Tape checksum error.

2. Attempting to load data into non-existent or read-only memory.

EXAMPLES

Example 1: Read a file and load the data into memory 256 (decimal) bytes above the load addresses specified in the file.

.R100<cr>
.

4.4.10 WRITE HEX FILE COMMAND

FUNCTION

The Write Hex File (W) command allows a block of memory to be output, in either 8086 or 8080 hexadecimal object file format, to a paper tape punch.

FORMAT

W[X]<start address>,<end address>[,<exec address>]<cr>

OPERATION

1. To use the Write Hex File command, enter W for 8086 file format or WX for 8080 file format and enter start address and end address of the memory block to be output. Note that no segment address value is permitted with end address (start address segment value is implied) and that if no segment value is specified with start address, the current CS register value is used. When the carriage return is entered, the following information is punched on the paper tape:

- Six inches of leader (60 null characters)
- An extended address record (8086 format only)
- The data contents of the memory block bounded by start address and end address (inclusive)

- An end-of-file (EOF) record
- Six inches of trailer (60 null character)

Optionally, an execution address can be specified prior to entering the carriage return. This is the memory address that is loaded into the CS and IP registers (IP register only with 8080 format) when the tape is read with the R command. Depending on the format selected, when execution address is specified, either an execution start address record containing execution address is punched immediately following the tape leader (8086 format) or the offset address value of execution address is punched in the EOF record (8080 format).

When using the 8086 format (W command), the start address segment value (CS register value if a segment value is not specified) is entered (punched) in the extended address record , and the start address offset value is entered in the load address field of the first data record. The segment and offset address values of execution address are entered in the execution start address record (CS register contents if a segment address value is not specified with execution address).

When using the 8080 format (WX command,), the start address offset value is punched in the load address field of the first data record. Execution address, if specified, is punched in the EOF record. Note that a segment address value is not permitted with execution address or end address and that the start address segment value is used only to define the starting address of the memory block and that it is not punched on the tape.

The Write Hex File command can be cancelled or stopped at any time by entering control characters from the console device. Control-C cancels the command and prompts for new command entry. Control-S stops the output, but does not cancel the command. Control-Q resumes output that has been stopped. The only console input allowed following a Control-S is either a Control-Q or a Control-C.

Additional information regarding Intel object file formats is available in the MCS 80/85 Absolute Object File Formats Technical Specification.

ERROR CONDITIONS

1. Specifying a value for end address that is less than the offset value of start address.

Examples

Example 1: Output the memory block bounded by 04H and 06DDH, relative to the current CS register, to an 8086 file with an execution address of CS:040H

```
.W4,6DD,40<cr>
.
```

Example 2: Output the memory block bounded by FF200H and FF2FFH to an 8080 file with a starting load address of 0100H and an execution address of 011AH

```
.WX FF10:100,1FF,11A<cr>
```

CHAPTER 5

HARDWARE

This chapter gives the hardware design details of ESA 86/88-2. Appendix A gives the complete schematics, Appendix B gives the connector details and Appendix C has the component layout diagram. The design details are discussed in the following order :

a) CPU
b) Memory addressing
c) I/O addressing and Peripheral clock
d) Wait State logic
e) Interrupt system
f) 8087 Co-processor (NDP)
g) Bus expansion
h) Connector details

5.1 CPU

ESA 86/88-2 can work either with 8086 CPU or with 8088 CPU. In either case, the CPU is configured for operation in the maximum mode.

5.1.1 CPU Reset

An on-board RESET key can provide a RESETIN* signal to the clock generator 8284A. The RESET output from 8284A is used to reset the CPU and rest of the system. This signal is available on the bus connector also and can be used to reset the off-board peripherals.

5.1.2 CPU CLOCK

A 24 MHz crystal oscillator is the clock source for 8284A clock generator. 8284A divides this frequency by three and produces a 5 MHz clock with 33 % duty cycle as required by 8086/8088. Further, 8284A provides a 4 MHz PCLOCK with 50% duty cycle. Both these clock outputs are available on the bus connector.

5.1.3 CPU ADDRESS BUS

Latches (74LS 373s at U22, U21 and U20 are used to latch the address with the help of ALE signal. These latches are always enabled since there is no provision for external bus master. Also note that BHE*, status signals S0*, S1* and S2* are also latched.

As the CPU operates in maximum mode, the 8288 bus controller directly decodes the status signals to provide all the control signals.

5.1.4. Corrected BHE*

8086 processor generates a BHE* signal on pin 34 and this signal is used to access the devices residing on the upper data bus. However, 8088 processor always drives a high signal on pin 34 as its access is always over the lower data bus. To accommodate this difference, a corrected BHE* signal is used in rest of the system. This corrected BHE* signal is either the latched BHE* signal (for 8086) or the inverted latched A0 signal (for 8088).

This selection is accomplished by tristate buffer 74LS125 at U41, the enable signals for which are derived from the setting of SW4 of the DIP switch. (Recall from chapter 2 that this switch setting indicates the selection of 8086/8088 CPU). Thus if 8086 is selected the LS125 will inhibit inverted A0 and pass latched BHE* signal as corrected BHE* signal. On the other hand, if 8088 is selected, the LS 125 will inhibit latched BHE* (which will be always high) and pass the inverted A0 signal as the corrected BHE* signal.

5.1.5 CPU DATA BUS

Bidirectional buffers (74LS245s) at U18, U16 and U17 are used to buffer the CPU data bus. Note that one buffer (74LS245 at U17) is used as swap buffer to route the upper data bus to lower data bus when 8088 is selected. The enable signals for the bus are derived from corrected BHE*, A0, DEN* (from 8288 bus controller) and 86 Sel* and 88 sel signals.

This logic implemented by 74LS32 at U40 is as follows:

CPU	**DEN ***	**A0**	**CORRECTED BHE***	**ENABLE FOR**		
				Upper data bus buffer	Lower data bus buffer	Swap buffer
8086	HIGH	X	X	NO	NO	NO
	LOW	LOW	LOW	YES	YES	NO
	LOW	LOW	HIGH	NO	YES	NO
	LOW	HIGH	LOW	YES	NO	NO
8088	HIGH	X	X	NO	NO	NO
	LOW	LOW	HIGH	NO	YES	NO
	LOW	HIGH	LOW	NO	NO	YES

(X : Dont Care

The combinations not shown above never occur)

5.2 Memory addressing

ESA 86/88-2 has four 28-pin JEDEC compatible slots (U2,U3,U4,U5) dedicated to PROM. These sockets can be collectively configured for 27128, 27256 or 27512. The total PROM capacity would be respectively 64K, 128K or 256K bytes. Two of the sockets (U2 and U3) are used by the system and the other two sockets (U4 and U5) are available to the user for expansion. For details regarding PROM configuration, refer to section 2.1.5. on Memory selection.

ESA 86/88-2 provides on-board static RAM of 128K bytes (4 X 32K) via four 28-pin JEDEC compatible slots U6,U7,U8 and U9. Two of these sockets are usually populated to give 64K Bytes of RAM and the other two sockets are available to the user for expansion. The RAM area starts from 0:0000H. The area from 0:0000 to 0:1FFFH is used by the system for interrupt vectors, stack and Assembler data tables. Thus user RAM starts from 0:2000H.

All the on-board memory is accessed with Zero wait states.

5.2.1. Memory Map:

PROM Area:

PROMS Selected	Device No.	Address Range (in Hex)
27128	U4 and U5	F0000 - F7FFF
	U2 and U3	F8000 - FFFFF
27256	U4 and U5	E0000 - EFFFF
	U2 and U3	F0000 - FFFFF
27512	U4 and U5	C0000 - DFFFF
	U2 and U3	E0000 - FFFFF

RAM area:

Device No.	Address range (in Hex)
U8 and U9	00000 - 0FFFF
U6 and U7	10000 - 1FFFF

Optional battery backup provision is available for U6,U7,U8,U9 (RAMs). Use 1.2V X 2 Ni.Cd. cells and connect them as per the polarity at the battery terminal points (Two berg pins) named "BT". Refer Appendix C for Component layout diagram.

5.3 Peripheral clock

A separate 6.144 MHz Crystal is used to generate peripheral clocks of 3.072 MHz and 1.536 MHz frequencies. The 3.072MHz clock is used for the 8251A USART and 1.536 MHz clock is used for the 8253-5 Programmable interval timer. The 3.072 MHz clock is available on the bus connector.

5.4 I/O Addressing and I/O Map

I/O decoding is implemented using U23 and U24. This section describes I/O devices provided on ESA 86/88-2.

5.4.1. Parallel I/O ports.

ESA 86/88-2 provides two 8255A programmable peripheral interface devices. One 8255A, at position U27, resides on the lower data bus and the other 8255A, at position U32, resides on the upper data bus. Each 8255A consists of three 8-bit input/output data ports (designated as ports A, B and C) and one write-only control port. All the ports can be addressed individually (eg. Port A of 8255 at U32) or corresponding pairs of ports (eg. Port A of 8255 at U27 and Port A of 8255 at U32) can be addressed as one 16-bit wide port.

The I/O address assignment can be found in the I/O map given in section 5.4.6.

The signals of the Port A, Port B and Port C of the 8255A at U27 are brought out to the connector J4 and the corresponding signals of the other 8255A at U32 are brought out to the connector J5. The pin assignments for the individual port bits are given in the last section of this chapter.

NOTE: Port bits PC0, PC3 of U27 and PC3 of U32 can act as interrupt sources allowing the operation of 8255As in Mode 3.(Ref.Section 2.1.6)

5.4.2 Onboard Keyboard/Display

The I/O address assignment for 8279 can be found in the I/O map given in section 5.4.6. The 8279 is configured for the following mode of operation:

- 8 digits, 8-bit, left entry display
- Encoded scan keyboard with 2-key lock out

The keyboard monitor reads the keyboard by polling the status port of 8279. The codes assigned to various keys of the keyboard are as follows:

Sl. No.	Key	Corresponding Code
1	0	00H
2	1	01H
3	2	02H
4	3	03H
5	4	04H
6	5	05H
7	6	06H
8	7	07H
9	8	08H
10	9	09H
11	A	0AH
12	B	0BH
13	C	0CH
14	D	0DH
15	E	0EH
16	F	0FH
17	.	10H
18	,	11H
19	-	12H
20	+	13H
21	:	14H
22	REG	15H
23	F1 (BLANK CHK)	16H
24	F2 (READ)	17H
25	F3 (PROGRAM)	18H
26	F4 (VERIFY)	19H
27	PREV	1AH
28	F5 (TAPE RD)	1BH
29	F6 (TAPE WR)	1CH
30	F7	1DH

NOTE : RESET and KBINTR keys are not connected to 8279 .
The signal generated by pressing the RESET key is used to drive the RESETIN* signal on the main system and thus can be used to reset the complete system.

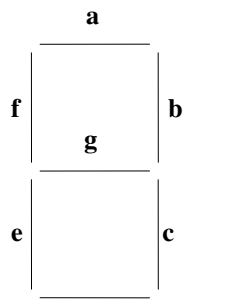
The signal generated by pressing the KBINTR key is used to generate an NMI signal to the CPU. The NMI service routine saves all the registers and returns to the command prompt mode. Thus, this key can

be used to recover gracefully from “stuck” programs. This key works correctly in both keyboard mode of operation and in serial mode of operation.

The function keys F1 through F7 can be redefined by the user if desired. Refer chapter 3 for details on the use of function keys.

Display Drive:

The segment drive outputs of 8279 (A0 through A3 and B0 through B3) from a single 8-bit parallel output driving the display element segments. The correlation between the individual bits and the display segments is as follows:



CPU DATA BUS D15 D14 D13 D12 D11 D10 D9 D8

8279 OUTPUT BITS A3 A2 A1 A0 B3 B2 B1 B0

SEGMENT CONTROLLED dp g f e d c b a

Data Bit = 1 Corresponding Segment is ON
0 Corresponding segment is OFF

Display Segments

Example: To dispaly "E", segments a,f,g,e,d should be ON and other segments should be OFF. So, the data format would be 01111001 = 79H. Display codes for other patterns can be worked out similarly.

5.4.3. Programmable Interval Timer

ESA 86/88-2 has an on-board programmable interval timer 8253 at position U12. 8253 has one command port and three data ports called Timer 0, Timer 1 and Timer 2. These provide three programmable timers. One of these, Timer 0 is utilized by the system to generate the baud rate clock for the 8251A USART. The outputs of the other two timers, along with their clock and gate inputs are available on the connector J6. Refer to the connector details at the end of this chapter.

The I/O address assignments can be found in section 5.4.6.

NOTE: The outputs from Timer 1 and Timer 2 can be used as on-board interrupt sources. Refer section 2.1.6.

5.4.4. Serial Interface controller

An 8251A USART at position U13, is used for implementing RS 232 C compatible serial interface. 8251A is programmed for asynchronous operation, 2 stop bits, no parity. The baud rate clock is provided by 8253 as explained in the previous section.

The I/O address assignments can be found in section 5.4.6.

NOTE: 8251A is operated by polling its status register. However its TxRdy and RxRdy outputs can act as on-board interrupt sources if required. Refer section 2.1.6.

5.4.5 Programmable Interrupt controller

ESA 86/88-2 has an on-board Programmable Interrupt controller 8259A at position U1. The INTR output of this device is connected to the INT input of the CPU. Interrupt inputs to 8259A can be on-board sources or off-board sources. Ref Section 2.1.6. for details of interrupt source selections.

The I/O address assignments can be found in the next section.

NOTE: The system firmware does not initialize the 8259A to any mode. Thus user program must configure this device appropriately before using it.

5.4.6 I/O Address Map

The following table summarizes the I/O mapped I/O address assignments and usage of all the peripherals of ESA 86/88-2. (Ref Component Layout Diagram in Appendix C).

Peripheral Device	Port Addr.	Device Register	Usage
DIP Switch	FFED		System configuration
8253 (PIT)	FFFF	Command port	
	FFF9	TIMER 0	Used for baud rate generation
	FFFB	TIMER 1	available to user
	FFFD	TIMER 2	available to user
8251(USART)	FFF2	Command/status port	used for serial communication
	FFF0	Data port	
8255(PPI)	FFE7	High control port	available to user
High, at U32	FFE1	High port A	
	FFE3	High port B	
	FFE5	High port C	
8255(PP1)	FFE6	Low control port	
Low, at U27	FFE0	Low port A	available to user
	FFE2	Low port B	
	FFE4	Low port C	
8259A(PIC)	FFF4	Command/status	available to user
	FFF6	Data port	
8279(KB/DC)	FFEB	Command/status	Keyboard/Display
	FFE9	Data port	Controller

5.5 Wait State Logic

As noted earlier, all memory cycles are run with zero wait states. All I/O cycle are executed with one wait state. The wait state logic is implemented by a 74 LS 76 at U38. Whenever, an I/O cycle is in progress, the D input to LS76 (=LS2) becomes high and the Q* output of LS76 goes to zero on the raising edge of T2 state de asserting the RDY1 signal to 8284A. On the next low-high transition of the clock, the Q* output toggles again, reasserting the RDY1 signal to 8284A. Thus one wait state is introduced whenever an I/O cycle is executed.

The preset input, pulled up to VCC, can be controlled by off-board devices (memory or I/O) to introduce as many wait states as required. This line, designated as EXTRDY is available on the system connector J1. When this line is pulled low, ready input to 8284A is de aserted and when this line is high or open there is no influence on the ready line. This scheme permits the off-board devices to introduce wait states if required or operate with one wait state for I/O cycles.

5.6 Interrupt System

5.6.1 Hardware Interrupts : The 8086/8088 CPU has two hardware interrupt inputs, called NMI and INT.

The NMI (Non Maskable Interrupt), as already mentioned is connected to the KBINTR key of the Keyboard/Display module. The NMI service routine saves all the CPU registers and returns the monitor to the command prompt mode.

The INT (maskable interrupt) is connected to the INTR output of 8259A. Ref section 2.1.6 for details of interrupt selections.

5.6.2 Internal Interrupts : INT3 can be used by user programs to return control monitor. Other internal interrupts are available to user.

5.7 8087 Numeric Data Processor

On-board provision is made for 8087 coprocessor. The CPU can be either 8086 or 8088 and to install 8087, no hardware changes are necessary. With power off, install the 8087 carefully into the socket U28 (Refer the component layout diagram in Appendix C).

The interrupt output from the NDP can be used as an on-board interrupt source (refer section 2.1.6)

5.8 BUS EXPANSION

The buffered address, data and control signals are provided on the bus connectors J1 and J2.

Refer to the connector details in the next section .

5.9 Connector Details.

There are in all six connectors J1 through J5 (refer the component layout diagram in Appendix C to locate these connectors). Of these connectors, J1 and J2 are 50 pin connectors and provide signals for bus expansion; J4 and J5 are 26 pin connectors and provide parallel I/O signals; J3 is a 25 pin, D-type, female connector used for RS 232 C compatible serial communication. The signal definitions on all these connectors are listed below (This information is available in Appendix B also).

J1-CONNECTED TO SYSTEM BUS

PIN NO	SIGNAL	PIN NO	SIGNAL
1	GND	2	GND
3	+5V	4	+5V
5	+5V	6	+5V
7	GND	8	GND
9	GATE1	10	GATE2
11	CONCLK	12	RESET
13	N.C	14	PCLK
15	CINT0*	16	CINT1*
17	CINT2*	18	CINT3*
19	CINT4*	20	CINT5*
21	CINT6*	22	CINT7*
23	LS0	24	INTA*
25	LS1	26	MWTC*
27	LS2	28	IOWC*
29	NC	30	CAS0
31	NC	32	CAS1
33	LOCK*	34	CAS2
35	MRDC*	36	AMWTC*
37	IORC*	38	AIOWC*
39	ALE	40	EXTRDY
41	CLK1	42	CLK2
43	GND	44	GND
45	+5V	46	+5V
47	TIMER1	48	TIMER2
49	GND	50	GND

J2-CONNECTED TO SYSTEM BUS

PIN NO	SIGNAL	PIN NO	SIGNAL
1	GND	2	GND
3	BA0	4	CBHE *
5	BA1	6	BA2
7	BA3	8	BA4
9	BA5	10	BA6
11	BA7	12	BA8
13	GND	14	BA9
15	BA10	16	BA11
17	BA12	18	BA13
19	BA14	20	BA15

21	BA16	22	GND
23	BA17	24	BA18
25	BA19	26	NC
27	NC	28	CRTC
29	NC	30	GND
31	BD0	32	BD1
33	BD2	34	BD3
35	BD4	36	BD5
37	BD6	38	BD7
39	GND	40	GND
41	BD8	42	BD9
43	BD10	44	BD11
45	BD12	46	BD13
47	BD14	48	BD15
49	GND	50	GND

J4 CONNECTED TO 8255 at U27 (ODD)

PIN No	SIGNAL	PIN No	SIGNAL
1	P1C4	2	P1C5
3	P1C2	4	P1C3
5	P1C0	6	P1C1
7	P1B6	8	P1B7
9	P1B4	10	P1B5
11	P1B2	12	P1B3
13	P1B0	14	P1B1
15	P1A6	16	P1A7
17	P1A4	18	P1A5
19	P1A2	20	P1A3
21	P1A0	22	P1A1
23	P1C6	24	P1C7
25	+5V	26	GND

J5 CONNECTED TO 8255 at U32 (EVEN)

PIN NO	SIGNAL	PIN NO	SIGNAL
1	P2C4	2	P2C5
3	P2C2	4	P2C3
5	P2C0	6	P2C1
7	P2B6	8	P2B7
9	P2B4	10	P2B5
11	P2B2	12	P2B3
13	P2B0	14	P2B1
15	P2A6	16	P2A7
17	P2A4	18	P2A5
19	P2A2	20	P2A3
21	P2A0	22	P2A1
23	P2C6	24	P2C7
25	+5V	26	GND

J3 SERIAL COMMUNICATION

(CONNECTOR 9 PIN 'D' TYPE FEMALE)

PIN NO	SIGNAL
1	NC
2	RxD
3	TxD
4	(pulled high)
5	GND
6	NC
7	RTS
8	CTS
9	NC

CHAPTER 6

MONITOR ROUTINES ACCESSIBLE TO USER

ESA 86/88-2 monitor offers several user-callable useful routines both in keyboard and serial modes of operation, details of which are given below. For programming examples using the same refer to chapter 8 on programming examples.

The following remarks apply to the use of any and all User-callable monitor routines:

a) All the CALLs are intersegment CALL instructions. (OP code 9AH).
b) Before making such an intersegment CALL, DS must have a value of 0 and stack must be able to accomodate at least 12H bytes.
c) The monitor routines are guaranteed to preserve the values of the following registers only:

1)CS 2)DS 3)SS 4)SP

6.1 KEYBOARD MONITOR ROUTINES ACCESSIBLE TO USER

Name	Functions	Calling Address	Parameters
1) UKB $ DISPLAY	Displays an array of characters	FF00: 0B31	Reg AL= Field of Display 1-use address field 0-use data field Reg AH = Number of prompts (dots 0,1,2 or 3; Reg ES, BX = Segment, offset of the starting address of the array to be displayed.
NOTE: For detail of display format see section 5.4.2 .			
2) UKB $ BLANK $ ADDR $ FIELD	Blanks the address field of the display with the number of prompts specified by the input parameter.	FF00: 0B05	Reg AL = Number of prompts 0,1,2 or 3.
3) UKB $ BLANK $ DATA $ FIELD	Blanks the data field of the display with the number of prompts specified by the input parameter.	FF00: 0B00	Reg AL = Number of prompts 0,1,2 or 3.
4) UKB $ $OUT $ BYTE	Outputs a byte to the data field with the number of prompts specified by	FF00: 0B0A	Reg AL = Byte value to be output Reg AH = Number of prompts 0,1,2 or 3.

	the input parameter.		
5) UKB $ OUT $ WORD	Outputs a word to the data field or address field with the specified number of prompts. Leading zero blanking is performed if so specified.	FF00: 0B12	Reg AX = word value to be output Reg BH=Display field 1 - Address field 0 - Data field Reg BL = No of prompts 0,1,2 or 3 Reg CL = Leading zero blanking. 1 - performed 0-Not perfomed.
6) UKB $ GET & CHAR This	Reads one character from keyboard	FF00: 0B1C	Input = None **Output parameters :** Character read from the keyboard is returned in location 0000: 005A. character is also available in register AL
7) UKB $ GET EXPRESSION	Forms a word or byte expression from the characters. read from the keyboard.	FF00: 0B20	Reg AL = Field to which characters are echoed 01 - Address field 00 - Data field (word input) FF - Data field (byteinput) Reg AH = Number of prompts - 0, 1, 2 or 3 Reg CL = Leading zero blanking 1-performed 0-Not performed **Output parameters:** Reg AX =value of the expression Location 0000:005A = Expression terminating character.

NOTE 1: Assumes the first character is already available in the location 0000:005A.

NOTE 2: Expressions must be terminated with a comma, period or colon. Otherwise the routine is terminated, error message is displayed and monitor enters the command entry mode.

8) UKB $GET $ ADDRESS	Forms an address expression from the characters read from the keyboard. The address expression consists of an optional segment value and the offset value, separated by colon (when segment	FF00: 0B29	Reg ES,Reg BX=Segment and offset value of the pointer location into which the address value gathered from keyboard is placed Reg AX=Default value for the segment value of the address expression to be used when the user does not enter a segment value.

value is entered). The expression is displayed in the address field.

Reg CL = Number of prompts 0,1,2 and 3.

Output parameters: address value gathered from keyboard, returned in the location specified by two input parameters. Location the first 0000:005A = Expression terminator -a comma or a period.

NOTE 1: Assumes that the first character of the expression is already available in the location 0000:005A.

NOTE 2: Address expressions must be terminated with a comma or a period. If not, the routine is aborted, error message is displayed and monitor returns to command entry mode.

6.2 SERIAL MONITOR ROUTINES ACCESSIBLE TO THE USER

Name	Functions	Calling Address	Parameters
1) US $ OUT $ CHAR	Output an ASCII character to the console	FE00: 1B50	Input : (AL) = ASCII character
2) US $ OUT $ STRING	Output a string of ASCII characters to the console. Assumes the string is terminated by ASCII NULL (=00H)	FE00: 1B55	Input:(ES) = segment where string is located (AX) = offset of string
3) US $ OUT $ CRLF	Output a carriage return and Line feed to the console	FE00: 1B5B	None
4) US $ OUT $ BLANK	Output a blank character (ASCII code-20H) to the console	FE00: 1B60	None
5) US $ OUT $ BYTE	Output a byte value to the console	FE00: 1B64	Input: (AL) = Byte (in Binary) value to be output
6) US $ OUT $ WORD	Output a word value to the console	FE00: 1B69	Input: (AX) = word (in Binary) value to be output
7) US $ OUT $ ADDR	Output an address value in the form segment value :	FE00: 1BA9	Input: (ES) = segment (in Binary) value (BX)=offset value; to

	offset value to the console.		be output on the console
8) US $ GET $ CHAR	Read a character from the terminal keyboard	FE00: 1B70	Inputs: None Output: (AL) = ASCII code of key pressed
9) US $VALID $ HEX	Check if a character from the terminal is a valid hex character (‘0’ to hex, ‘9’, ‘A’ to ‘F’)	FE00: 1B77	Inputs: (AH)= ASCII character Output: (AL) = FF if valid, 00 otherwise
10) US $ HEX	Convert a valid hex character in ASCII into equivalent binary number	FE00: 1B80	Inputs: (AL)= Valid hex character in ASCII Outputs : (AL) = Binary equivalent
11) US $ GET $ WORD	Get a word value from the terminal keyboard (Note: if the terminator is not a CR or `:' or ‘,’ control returns to monitor)	FE00: 1B85	Input: None output: (AX) = Word Input (CL) = Terminator (CR or : or ,)
12) US $ GET $ WORD$C	Same as US$ GET $ WORD except that the first character is assumed to be already avail- (CR or : or ,) able in location 0:0066.	FE00: 1B88	Input: None Output: (AX) = Word Input (CL) = Terminator
13) US $ GET $ ADDR	Get an address value (segment value :off set value) from the terminal keyboard. Allows the specification of a default segment value. (Note: If the terminator is not a CR or `,' control returns to monitor)	FE00: 1B90	Inputs: (ES), (BX) = segment and offset of the pointer which is to hold the address gathered from key board. (AX) = Default value for segment register which is to be used if user does not enter any segment value. Outputs: Address gathered from keyboard placed in the pointer location. specified (CL) = Terminator (CR or ,)
14) US $ GET $ ADDR$C	Same as US $ GET $ ADDR except that the first character is assumed to be already available in memory location 0:0066.	FE00: 1B9E	Same as for US $ GET $ ADDR

CHAPTER 7

ESAM – 86
SYMBOLIC ASSEMBLER

7.1 INTRODUCTION:

ESAM-86, the symbolic one line assembler provided with the ESA-86/88-2 is capable of translating the mnemonic instruction codes into machine codes. The translated code is immediately loaded into appropriate memory locations. This assembler, ESAM-86, supports the standard 8086/8088 mnemonics with some minor easy to understand modifications. This assembler supports labels (Symbolic references) also. These lables consists of one or two characters. However a label can be referenced only if it has already been defined. In other words ESAM-86 supports backward references only.

In addition to the standard instruction mnemonics, ESAM-86 provides some useful Assembler directives (Pseudo opcodes). These directives can be used to set the origin, define symbolic constants, initialise byte, word and string values, and to reserve memory space.

Further, ESAM-86 provides a disassembly facility also. Using this facility, user can see the codes in standard mnemonic form and verify them.

ESAM-86 occupies 16K bytes of code space. It requires 6.5K bytes of RAM area, from the location 0100H to 19FFH.

NOTE:

- During the assembly process, assembler makes use of ESC A sequence to move the cursor up by one line. Make sure that your terminal supports this function in similar fashion. If not, results could be unpredictable.

- While using the Assembler the user must enter his/her program starting from RAM location 2000H. Entering program in reserved locations will lead to unpredictable results.

7.1.1 INSTALLATION

ESAM-86 Assembler is a built-in facility and thus no installation procedures are necessary .

7.2. OPERATION

7.2.1 INVOKING THE SINGLE LINE ASSEMBLER ESAM-86

The single line Assembler ESAM-86 is invoked from the serial monitor. When the serial monitor prompts for a command with the character '.', enter the command mnemonic A (for Assembler). Now the monitor will transfer control to the Assembler.

When the assembler ESAM-86 gains control of the system, it will display the sign-on message “ESAM-86 SYMBOLIC ASSEMBLER V x.y” and issue its own command prompt character (a “>”) on the next line. Notice that the sign-on message would be the same with either 8086 CPU or 8088 CPU. The mnemonics for both the processors are identical ! Now the user can enter any of the commands supported by ESAM-86. These commands are described in detail in later sections .

7.2.2 EXITING FROM ESAM-86

To exit from the ESAM-86, enter EX followed by CR, when prompted for a command (with the prompt “>”). Now, control of the system returns to the serial monitor which will issue the sign-on message and the command prompt ‘.’.

The user can invoke the ESAM-86 and exit from it any number of times

7.2.3 STRUCTURE OF ESAM-86 COMMANDS

Whenever ESAM-86 is ready to accept a command from the user, it will output the character ">" as the command prompt at the beginning of a new line.

The commands entered by the user consist of a two character command mnemonic followed by a list of command parameters. This list may consist of zero to three parameters depending on the particular command being used. When more than one parameter is required, a single comma (',') is used between the parameters as a separator.(In many contexts, a space is also accepted as a seperator.) A command is terminated by the carriage return. Commands are executed one at a time and only one command is allowed with in a command line.

Any command is interpreted only after the Carriage Return is entered. Thus user can correct entry errors using the Backspace (CNTL-H) key.

The following are the command mnemonics supported by ESAM-86.

SG	(Set Code Segment Register)
LC	(Label clear)
DA	(Disassemble and Assemble)
DI	(Disassemble)
EX	(Exit from ESAM-86)

The commands are described in detail in the following sections.

***Parameter Entry*:**

As in the case of the serial monitor, hexadecimal number system is used. However for byte entry a maximum of 2 characters, and for word entry a maximum of 4 characters are to be entered. However note that all numbers must begin with numerical. Thus, a leading zero must be used if the number starts with A to F. For eg, ACH must be entered as 0ACH

Response to Errors:
When an error is detected during command entry, the command is aborted, appropriate error message is issued and command prompt character ('>') is output at the beginning of a new line.

7.2.4 DESCRIPTION OF ESAM-86 COMMANDS:

7.2.4.1 SG COMMAND

SG <value><CR>

This command is used to set the value of the code segment register. Once this value is set, it remains valid for the current session.

NOTE:
At the beginning of a fresh session, initialize the value of code segment since it is not initialized to any default value.

7.2.4.2 LC COMMAND:

LC <CR>

This command will clear the symbol table and any label definitions generated in the earlier invocations of the assembler are lost. Thus this command is used when fresh assembly of a source program is being done.

If this command is not used, earlier label definitions will remain valid for this session. If the user is patching an already assembled program, he may wish to use the labels already defined. In such cases he / she can retain those definitions by not using the command.

The detailed syntax regarding the definition and use of labels is described in the next section on Assembly language conventions.

NOTE:
Use LC Command to clear the label tables on entering the Assembler for the first time.

7.2.4.3 DA COMMAND:

DA <Addr><CR>

This command is used to disassemble machine code and assemble source statements.

Upon receipt of this command, ESAM-86 prints the address value followed by the code and disassembled instruction at this address. Now it waits for input of source statement from the user. At this point, user has three options:

1. He can enter a source statement (The detailed syntax governing source statements is described in the next section on Assembly language conventions). or
2. He can enter the carriage return if he doesn't wish to change this instruction. Or
3. He can enter the character '!' followed by CR to terminate the command

If the user enters a new source statement, it is immediately translated. If any errors are detected, appropriate error message is issued in the mnemonic field on next line and location counter is not updated and the old line is displayed again. If there are no errors, the assembled machine code is immediately loaded into the memory, the valid source statement and its codes overwrite the previous Mnemonic and codes and the location counter is updated. The instruction at the new value of the location counter is disassembled and displayed along with the codes on the next line and the process continues.

NOTE:
The source line entered by the user is analyzed only after a carriage return is entered. Thus user can correct the entry errors using the Backspace (CNTL-H) key.

If the user doesn't wish to change the displayed instruction he can enter the carriage return. The location counter will be updated, the instruction at the new value of the location counter is disassembled and displayed along with the codes on the next line. And the process continues.

If the user enters '!' followed by CR then XT 86 environment is saved and the DOS command is executed Then XT 86 environment is restored and XT 86 command prompt is displayed again.

Note:
Do not "attach" the printer when using this command since the disassembled line will be overwritten by a valid user source statement

EXAMPLE:

```
        >DA 2000 <cr>

2000    FB                      STI ; NOP

2000    90                      NOP;

2001    90                      XCHGW AX,AX;POP CX
                                ILLEGAL OPERAND;

2001    90                      XCHGW AX,AX; POP CX

2001    59                      POP CX;

2002    90                      XCHGW AX,AX;A2 MOVB AL,BL

2002    88  00      A2          MOVB AL,BL;

2004    59                      POP CX;A456 CLD

2004    FC          A4          56 CLD

2005    FB                      STI; JMP QW
                                ILLEGAL OPERAND;

2005    FB                      STI; JMP A2

2005    EB  FB                  JMP A2

2007    FF                      ILLEGAL OPCODE;!
>
```

NOTE:

1) Underlined ones indicate that are overwritten over the original lines.

2) Error message output because in the second line there in no space after ";" (It must be a space if there is no label).

3) The assembler stores only the first two characters of a label. Hence the indication at address 2004.

4) Label (QW) not previously defined. so error

5) If the code in the addressed location is not a valid 8086/8088 code, then it is disassembled as illegal opcode.

7.2.4.4 DI COMMAND:

DI <ADDR1> <ADDR2> <cr>

When this command is issued the codes from address 1 to address 2 are disassembled and displayed along with the machine codes on the console (Note: Addr1,Addr2 are offset values only. The segment value is always set using the 'SG' command)

EXAMPLE

```
>DI      2000, 2007        <cr>
2000      90                          NOP;
2001      59                          POP CX;
2002      88 00                       MOV AL, BL
2004      FC                          CLD
2005      EB FB                       JMP 2002
2007      FF                          ILLEGAL OPCODE;
>
```

NOTE:

To obtain a hard copy of a program in disassembled form, "attach" the printer and give `DI' Command.

7.3 ASSEMBLY LANGUAGE CONVENTIONS:

Each source line can be upto a maximum of 80 characters terminated by a carriage Return. Each line can contain one or more of the following fields.

[Label]...<Mnemonic>.....[operand (s)]

[...] optional field
<...> required field
..... one or more space

The different fields in a source line are identified by the order in which they appear. These fields are separated by one or more blanks acting as the delimiters. A comma may also be used as a delimiter.

7.3.1 LABEL FIELD:

The label field consists of either a space (when there is no label) or a label of one or two character. The first label character must be an alphabetic character. The second character if present can be an alphanumeric. If more than two characters are assigned to the label, the assembler stores only the first two characters and ignores the rest of the characters. Hence if more than two characters are assigned to a label, the assembler during assembly, outputs the first two characters in label field and the others in mnemonic field if the source statement is valid.

Thus the label, if present, starts from the first character of the source statement. And when there is no label, the first character must be a space.

NOTE:

The following reserved words cannot be used as labels

AH, AL, AX, CS, DS, SS, ES, CH, CL, CX, BH, BL, BX, SP, BP, SI, DI, DH, DL, DX

7.3.2 MNEMONIC FIELD:

The next field is the mnemonic field. This field always begins with an alphabetic character and may consist of upto 50 characters. No embedded blanks are allowed. Also, this field is used for Directives, (Pseudo opcodes) such as EQU.

The assembler supports the standard INTEL mnemonics with some easy to understand modifications. Thus, for example, to differentiate between BYTE and WORD operations, the basic mnemonic is suffixed with a `B', indicating a BYTE operation or with a `W' indicating a WORD operation.

Eg:

The instruction to increment register AL will be "INCB AL"

The instruction to increment register AX will be "INCW AX"

7.3.2.1 SEGMENT OVERRIDE OPTION:

Segment override option can be exercised by giving the Segment Register instruction prior to the actual instruction eg. "MOV ES: (SI), AL "instruction specifies a byte movement with the segment register ES overriding the default segment register DS. To realize this instruction in the present assembler, you must enter the following statements:

ES <cr>

MOVB [SI],AL

NOTE:
The segment Register override instruction will be applicable for all the instructions that follow it. The assembler will not check for the override capability of the next instruction. Thus if you want the override

option for only one instruction, you must restore the default segment register explicitly. All the instruction mnemonics are summarized in section 7.3.5.

7.3.2.2 DIRECTIVES (PSEUDO OPCODES):

In addition to the normal opcodes which generate executable machine instructions the Assembler recognizes the following Pseudo opcodes which occupy the mnemonic field like the normal opcodes.

. Origin control (ORG)	**.Word Initialization (DW)**
. Byte Initialization(DB)	**.Storage reservation (DSP)**
. Symbloic constant	**.String constant initialization**
definition (EQU)	**(ASC)**

Each of these directives is now discussed briefly.

1) Origin control (ORG)

ORG <Value> <CR>

ORG <Value 1> , <Value 2> <CR>

The location counter can be set to a specific value by the use of this directive.

It will generally be used as the first program entry for the starting location of the assembled code. For example ORG 2000H will result in the loading of next instruction assembled to start from location 2000 H.

The second form of the ORG directive is used to set the location counter as well as the segment value. (Equivalent to giving the SG command followed by origin setting.)

eg:

ORG 200 00 <CR>

will make (CS) reg. value as 200 H and location counter is set to 0.

2) Symbolic Constant Definition (EQU)

User can define a value for a symbolic constant by using the equate (EQU) directive

<label> EQU <defined constant> <CR>

For example, the symbol T1 can be defined to be equal to 2000 H, by entering T1 EQU 2000.

Further the EQU directive allows the user to assign the value of another symbol. For example, if T1 is already defined as 2000 H, then entering T2 EQU T1 defines T2 also as 2000 H <CR>.

Examples:

T1 EQU 2000 : T1 represents 2000 H
T2 EQU T1 : T2 represents 2000 H
Y EQU 10 : Y represents 10 H

3) Storage Reservation (DSP)

DSP <value> <CR>

User can reserve certain storage area without initializing it, by the use of this command. The location counter is updated by the 'value'. This can be done in both data and program area.

eg.

Assume current value of the location counter is 2360 H. Then the directive

DSP 20 <CR>

will increment the location counter to 2380 H. The locations 2360 H to 237F H are uninitialized.

4) Byte Initialization (DB)

<label> DB <value> [, <value>] <CR>

User can initialize a memory location to a particular value by the use of this command.This facility is particularly useful for entering a table of data as part of a program.

This directive supports upto two constants separated by a comma or space Eg. Assume the current value of the location counter is 2240 H.

Then the directive

T1 DB 12, 0A4 <CR>

will initialize location 2240 with the value 12H and the location 2241 with the value A4H. Further the symbol T1 is made equal to 2240H.

5) Word Initialization (DW)

<label> DW <value> <CR>

This directive is similar to DB directive except that, now a word of memory rather than a byte, is initialized. Further, only one value is permitted with one statement.

Eg.

Assume the value of location counter is 2240 H Then the directive

TY DW 1240 <CR>

Will initialize the word starting at 2240 H with the value 1240 H.

6) String constant Initialization (ASC) ASC 'String '

This directive allows the user to enter a string of characters and have these characters translated to ASCII Codes and stored in the memory. The string can have a maximum of 40 characters only. The string must be enclosed by a pair of single quote (') characters or by a pair of tilda (^) characters.

Eg.

Assume the value of the Location Counter is 2240 H. Now

ASC `ESA' <CR>

will initialize locations 2240H, 2241H and 2242H respectively with 45H, 53H and 41H
NOTE:

The assembler in this case will overwrite the original line, with the string displayed in opcode field

7.3.3 OPERAND FIELD:

After the Label and Mnemonic fields, we have the operand field. This field holds the operands as required by the particular instruction. This assembler supports all the standard addressing modes of 8086/8088. Section 7.3.5 summarises the notation for all the addressing modes.

When a numerical value is expected in the operand field, the user must supply it in hexadecimal notation. Further numbers should always start with a digit. Thus the value A5H should entered as 0A5. (Note that the suffix H is not entered).

When a label is used in the operand field, it must have been already defined. (i.e. forward references are not allowed)

7.3.4: ADDRESSING MODES

The ESAM-86 assembler supports all the addressing modes of 8086/8088 The syntax to be followed for the various addressing modes is as follows:

Base Registers	(BX)	or	(BP)
Index Registers	(SI)	or	(DI)
Base + Displacement	(BX) or (BP)	or	(SI) or (DI) preceded by value or label.
Base + Index	(BX)	(SI)	or
	(BX)	(DI)	or
	(BP)	(SI)	or
	(BP)	(DI)	

Immediate data	# VALUE # LABEL	or			
Displacement + Base + Index	VALUE LABEL	(BX) (BX)	(SI) (SI)	or	

NOTE:

1. All numerical values are to hexadecimal base.
2. Label when used, must have been defined in previous instructions.
3. Mnemonics are appended with a `B' or `W' depending on whether the operands are `bytes' or `words' respectively.

Section 7.3.5 can be referred for further details.

For IN and OUT instructions, it is not necessary to specify AL or AX. The format will be

INB or INW or OUTB or OUTW

The address of the port should be preceded by a colon (:)

Eg:

i.e INB **:** ADDRESS
or INB DX

7.3.5 MNEMONIC SYNTAX

This section gives the syntax of each Mnemonic.

NOTE:

1. A `B' or `W' is appended to the standard mnemonic, to indicate byte and word operations respectively.

2. The syntax for IN and OUT instructions differs the standards. See the table below.

INSTRUCTION		**OPERANDS**	**SYNTAX TO BE FOLLOWED**
1. AAA	ASCII adjust for addition	-	AAA
2. AAD	ASCII adjust for division	-	AAD
3. AAM	ASCII adjust for multiply	-	AAM
4. AAS	ASCII adjust for subtraction	-	AAS
5. ADC	ADD Destn Source with carry-byte mode.	EA(B),Reg, Reg,EA(B) AL,D8 EA(B),D8	ADCB

	ADD Destn & source with carry - word mode.	EA(W),Reg Reg,EA(W) AX,D16 EA(W),D16	ADCW
6. ADD	Add Destn & source Byte mode.	EA(B),Reg Reg,EA(B) AL,D8 EA(B),D8	ADDB
	Add Destn & source (word mode)	EA(W),Reg Reg,EA(W) AX,D16 EA(W),D16	ADDW
7. AND	And source Destn (Byte mode)	EA(B),Reg Reg,EA(B) AL,D8 EA(B),D8	ANDB
	And Source, Destn (Word mode)	EA(W),Reg Reg,EA(W) AX,D16 EA(W),D16	ANDW
8. CALL	Call to subroutine Indirect call	Disp 16 EA	CALL Address, or CALL Label. CALL (EA) CALL @ address
9. CALLS	Call interior Calls Segment routine	Drsp 16 or Seg 16 EA	CALLS disp 16, Seg 16 CALLS EA
10. CBW	Convert Byte to word	-	CBW
11. CLC	Clear carry flag	- -	CLC CLC
12. CLD	Clear direction flag	-	CLD
13. CLI	Clear interrupt flag	-	CLI
14. CMC	Complement carry flag	-	CMC
15. CMP	Compare Destn & source (Byte mode)	Regr, Regr Regr, EA(B) EA(B), Regr Regr,Immediate. EA(B),Immediate	CMPB

		AH,Immediate	
	Compare Destn & source (word mode)	Regr, Regr Regr, EA(W) Regr, immediate EA(B),immediate AX,Immediate	CMPW
16. CMPS	Compare destn & source strings (Byte mode) (Word mode)		CMPSB CMPSW
17. CS			CS
18. CWD			CWD
19. DAA	Decimal adjust for addn		DAA
20. DAS	Decimal adjust for subtraction		DAS
21. DEC	Decrement (Byte mode) Decrement (word mode)	Reg 8 EA(B) Reg 16 EA(W)	DEC B DEC W
DIV	Divide Unsigned (Byte mode) (Word mode)	EA(B) EA (W)	DIV B DIV W
DS			DS
ES			ES
ESC	External op-code source Escape		ESC
HLT	Halt	No operands	HLT
IDIV	Integer divide (Byte mode) Word mode	EA(B) EA(W)	I DIVB I DIVW
IMUL	Integer multiply Byte mode Word mode	EA(B) EA(W)	I MULB IMULW

IN	Input byte	Accumulator Port 8	 INB: Port 8
	Byte mode	Accumulator DX	 INB DX
	Word mode	Accumulator Port 8 Accumulator DX	 INW : Port 8 INW DX
INC	Increment Byte mode Word mode	EA(B) Regr,16 EA(W)	INCB INCW
INT	Interrupt type Type 3	Immediate	INT # DATA INT 3
INTO	Interrupt and over flow	-	INTO
IRET	Interrupt return	-	IRET
JA/JNBE	Jump if above Jump if not below or equal		
JAE/JNB/			Mnemonic same
JAE			
JB/JNAE			
JBE/JNA			
JC		Short label	
JCXZ			
JE/JZ			
JG/JNLE			
JGE/JNL		Short label	
JL/JNGE			
JLE/JNG			
JMP	Jump Intra segment		JMP address jmp Label JMP EA
JMPS	Indirect Jumps		JMP @ Address

JMPS	Jump Intersegment		JMPS disp 16, seg 16 JMPS EA
JNC			
JNE/JNZ			
JNO			
JNP/JNO			
JNS		Short label	
JO			
JPE/JP	Short label:	Short label is one whose displacement	
JS	is between locations +127 to -128 from the current value of IP.		
LAHF	Load AH from flags	-	LAHF
LDS	Load Pointer	Reg, EA(W) Using DS	LDS
LEA	Load effective address	Reg, EA(W)	LEA
LES	Load Pointer using ES	Reg, EA(W)	LES

* For the above 3 instructions mode II is not allowed.

LOCK	Lock bus	-	LOCK
LODS	Load string (Byte mode)	Source string	LODSB
	(Word mode)	"	LODSW
LOOP	LOOP	Short label	LOOP disp 8
LOOP/			
LOOPZ	Loop if equal or loop if zero	disp 8 bits	LOOPE disp 8
LOOPNE/			
LOOPNZ	Loop if not equal or not zero	disp 8	LOOPNE Disp 8

MOV	Move source destn	AH, addr 16	MOVB
		Addr 16, AH	
		EA (B), Reg	
		Reg, EA (B)	
		AH,	
		CH,	
		BL,	
		DL, D8	
		AH,	
		CH,	
		DH,	
		BH	
		EA(B), D8	
	(Word mode)	AX, addr 16	MOVW
		EA(W), Reg	
		Reg, EA(W)	
		AX, D16	
		CX	
		DX	
		BX	
		SP	
		BP	
		SI	
		DI	
		EA(W), D16	

* SR cannot be CS Register.

MOVS	Move String		MOVSB
	Byte mode		
	Word mode		MOVSW
MUL	Multiply unsigned	EA(B)	MULB
	Byte mode		
	Word mode	EA(W)	MULW
NEG	Negate	EA(B)	NEGB
	Byte mode		
	Word mode	EA(W)	NEGW
NOP*	No operation	No operands	NOP

* Actually coded as XCHGW AX, AX

NOT	Logical not		
	Byte mode	EA(B)	NOTB
	Word mode	EA(W)	NOTW
OR	Logical INCLUSIVE OR	EA(B), Reg	
	Src, Destn	Reg, EA(B)	
	Byte mode	AL, D8	ORB

	Word mode	EA (B) D8 EA, Reg Reg, EA(W) AX, D16	ORW
OUT	Output one port Byte mode Word mode	Immediate,AL DX, AL Immediate,AX DX,AX	OUTB :Port 8 OUTB DX OUTW: Port 8 OUTW DX
POP	pop destn word off stack	Regr, EA	POP
(POP CS is not allowed.)			
POPF	Pop flags off stack	-	POPF
PUSH	Push word onto stack	Regr EA	PUSH
PUSHF	Push flag near onto stack	-	PUSHF
RCL	Rotate left Through carry Byte mode word mode	EA(B),1 EA, CL EA(W),1 EA(W),CL	RCLB EA (B),1 RCLB EA(B) CL RCLW EA(W),1 RCLW EA(W),CL
RCR	Rotate right Through carry Byte mode Word mode	EA, 1 EA, CL EA(W), 1 EA (W), CL	RCRB EA(B),1 RCRB EA(B),CL RCRW EA(W), 1 RCRW EA(W), CL
REP/			REP
REPE/	Repeat Prefix		REPE
REPZ	(for string Instructions)		REPZ
REPNE/			REPNE
REPNZ	Repeat Prefix		REPNZ
RET	Return from subroutine (intra segment)		 RET no operand RET data 16

RETS	Return from subroutine (intersegment)	RETS no operand RETS data 16
ROL	Rotate left (Byte mode) Rotate left (Word Mode)	ROLB EA(B),1 ROLB EA(B),CL ROLW EA(W), 1 ROLW EA(W), CL
ROR	Rotate Right (Byte Mode) Rotate Right (Word Mode)	RORB EA(B), 1 RORB EA(B), CL RORW EA(W), 1 RORW EA(W), CL
SAHF	store AH in flags	SAHF
SAL/SHL	Arithmetic/logical Left shift (byte Mode) Word Mode	SALB EA, 1 SALB EA,CL SALW EA,1 SALW EA,CL
SAR	Arithmetic Right shift Byte Mode Word Mode	 SARB EA(B),1 SARB EA(B),CL SARW EA(W),1 SARW EA(W),CL
SBB	Integer subtraction with Borrow (Byte Mode) (word Mode)	SBBB EA(B),Reg SBBB Reg,EA(B) SBBB AL,Data8 SBBB EA (B),Data SBBW EA(W),Reg SBBW Reg,EA(W) SBBW AX,D16 SBBW EA(W),D8 SBBW EA(W),D16
SCAS	Byte Mode Word Mode	SCASB SCASW
SHR	Logical right shift (Byte mode) Word mode	SHRB EA,CL SHRW EA, 1 SHRW EA, CL
SS	Stack Segment over ride Prefix	 SS
STC	Set carry Flag	STD

STD	Set Direction Flag	STI
STOS	Byte Mode Word mode Store AX/AL into string	STOSB STOSW

SUB	Subtract
	Byte Mode
	
	Word mode
		SUBB EA(B),Reg
		SUBB Reg,EA(B)
		SUBB AL, D8
		SUBB EA(B), D8
		SUBW EA(W), Reg
		SUBW Reg, EA(W)
		SUBW AX D16
		SUBW EA (W), D16

TEST	Logical Compare Byte Mode Word Mode	 TestB EA, Reg AL,D8 EA,D8 TestW EA, Reg AX, D16 EA, D16
WAIT	Wait till TEST PIN not asserted	 WAIT
XCHG	Exchange Byte Mode Word Mode	 XCHGB Reg, EA(B) XCHGW Reg, EA(W) Reg, Reg
XLAT	Table look up Translation	XLAT
XOR	Logical inclusive OR Byte Mode Word Mode	XORB EA, Reg XORB REG, EA XORB AL, D8 XORB EA, D8 XORW EA, Reg XORW Reg, EA XORW AX, D16 XORW EA, D16

CHAPTER 8

PROGRAMMING EXAMPLES

8.1 INTRODUCTION

In this chapter, we will describe some programming examples which can be run on the ESA 86/88-2 system. These examples range from fairly simple ones designed to illustrate the use of various commands to fairly comprehensive examples designed to illustrate the use of monitor routines, testing and debugging techniques. All the programs except the ones illustrating the use of 8087, can be entered using the on-board assembler ESAM-86. In such a case, ensure that the program is entered correctly, by disassembling it after entry. Alternatively, the listed machine codes can be directly entered into the memory.

The user who is encountering the ESA 86/88-2 trainer for the first time is strongly urged to go through this chapter in detail, load and execute the programs as indicated. This will be of use in getting familiar with 8086/8088 programming as well as in getting to know the powerful features of ESA 86/88-2. The experienced user also might benefit from this chapter as it can make him/her familiar with the specific features of ESA 86/88-2 system.

All the examples of this chapter assume that the CPU installed is 8086. However, these programs can be run without any changes with 8088 CPU also. The only difference would be the sign-on message produced by the system upon power ON/RESET.

NOTE : User area of RAM starts from 2000H.

8.2 FAMILIARIZATION EXAMPLES:

The following simple examples are designed to familiarize the user with the operation of ESA 86/88-2 system.

Example 1: This example assumes that ESA 86/88-2 is operating in the keyboard mode (i.e. switch 4 of the on board DIP switch is in the ON position at power-ON.)

This program computes the average of given word values. The computed average is stored in given memory locations. Note that the following program does not check for overflow while forming the sum of the data values.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	29 C0		SUBW AX AX	;Clear the sum
2002	B9 04 00		MOVW CX #0004	;No. of elements
2005	BE 08 00		MOVW SI #0008	;Point subscript to end of table
2008	4E	AW	DECW SI	
2009	4E		DECW SI	;Back up one element
200A	03 84 1D 20		ADDW AX 201D[SI]	;Add element
200E	E2 F8		LOOP AW	;until CX=0
2010	29 D2		SUBW DX DX	;clear upper dividend

```
2012    B9 04 00            MOVW CX #0004     ;set up divisor ( = 4)
2015    F7 F1               DIVW CX           ;Divide the sum to get the average
2017    A3 1B 20            MOVW 201B AX      ;and store it
201A    CC                  INT 3             ;Return to monitor
201B    FF FF       RE      DW 0FFFF          ;space for result
201D    00 10       TA      DW 1000           ;Data
201F    00 20               DW 2000
2021    00 30               DW 3000
2023    00 40               DW 4000
```

1. Using the Examine Byte (EB) or Examine Word (EW) command enter the above program and data into locations 2000H to 2024H. For user convenience, the memory contents are listed below.

LOCATION	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2000	**29**	**C0**	**B9**	**04**	**00**	**BE**	**08**	**00**	**4E**	**4E**	**03**	**84**	**1D**	**20**	**E2**	**F8**
2010	**29**	**D2**	**B9**	**04**	**00**	**F7**	**F1**	**A3**	**1B**	**20**	**CC**	**FF**	**FF**	**00**	**10**	**00**
2020	**20**	**00**	**30**	**00**	**40**											

2. Using the GO command execute the program at location 2000H. After the program is executed, control returns to the monitor (because of the Breakpoint instruction INT 3).

3. Examine the contents of the word location RE(SULT) (201BH). It will be 2800 H, the average of the given four values 1000 H, 2000 H, 3000 H and 4000 H.

Suggestions

1. Try the program on different data. What happens when the table contains very large values (say A0A0H; FF22H; BCBCH; D4D4 H)? Incorporate a check for overflow.

2. Modify the program so that it computes the average of a set of byte values.

3. Modify the program so that

   (a) The number of data values is passed as parameter in CX register and
   (b) The offset address of TA(BLE) is passed as a parameter in BX register.

4. Rewrite the above program in 8085 code and compare the memory and execution time requirements.

NOTE:1. If you press RESET key, the contents of the user registers are not guaranteed to remain unaltered. The INT 3 instruction returns control to the monitor which will then save the complete user context. Thus all normal user programs will have INT 3 as the last instruction.

Example 2: This example also assumes that ESA 86/88-2 is operating in the keyboard mode.

The following program moves a block of data from one area of the memory to another area using the powerful string instructions of 8086/8088.

Let us assume that we want to move the data from 0:2100 through 0:210F to the block starting at 0300: 0

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	B8 00 03		MOVW AX #0300	;Set up ES and DI
2003	8E C0		MOVW ES AX	; to point to destination block.
2005	BF 00 00		MOVW DI #0000	
2008	BE 00 21		MOVW SI #2100	;Set up SI to point to source. (DS) is assumed to be 000 H.
200B	FC		CLD	;Clear direction flag for auto increment.
200C	B9 10 00		MOVW CX #0010	;count = 10 H
201F	F3		REP	
2010	A4		MOVSB	
2011	CC		INT 3	

1. Enter the desired data in locations 0:2100 to 0:210FH
2. Enter the above program into locations 0:2000 to 2011H
3. Execute the above program at 0:2000 using the GO command
4. Now the data would be moved to the block 0300:0 to 0300:0FH. You can examine this data using EB. (Examine Byte) command by giving the address as ES:DI (as this is how the address is specified in the program.) The required key sequence would be EB, reg, ES,:,reg, DI, NEXT, PREV, PREV, PREV,....

Example 3: This example assumes that ESA 86/88-2 is operating in the serial mode.

The following program converts a hexadecimal byte value to its ASCII notation. The example illustrates the use of the powerful translate (XLAT) and rotate instructions.

The program assumes that the hex value is in AL register. The resulting ASCII representation is left in the AX register.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	8D 1E 11 20		LEA BX 2011	;Get address of ASCII table
2004	30 E4		XORB AH AH	;Clear upper byte
2006	B1 04		MOVB CL #0004	;AX to be rotated by 4 bits
2008	D3 C8		RORW AX CL	
200A	D2 CC		RORB AH CL	;Lower nibble in AH and upper nibble in AL.
200C	D7		XLAT	;ASCII code of upper nibble in AL.
200D	86 E0		XCHGB AH AL	;Swap the two bytes
200F	D7		XLAT	;ASCII code of lower nibble in AL.
2010	CC		INT 3	
2011	30 31	AT	DB 30 31	
2013	32 33		DB 32 33	
2015	34 35		DB 34 35	
2017	36 37		DB 36 37	
2019	38 39		DB 38 39	
201B	41 42		DB 41 42	
201D	43 44		DB 43 44	
201F	45 46		DB 45 46	

1. Enter the above program and data into locations 2000H-2020H
2. using the Examine Register (XAX) command, enter a hex value say 2BH into AL register.
3. Using GO command execute the above program.
4. Using X command, examine the contents of AX register. It will be 3242H-ASCII representation of 2BH.

Suggestions

1. Try the above program for different hex values.
2. Write a program to convert ASCII code to EBCDIC code.

Example 4: This example also assumes that ESA 86/88-2 is operating in the serial mode.

This program searches through a byte array for a specified value. The parameters to be passed are as follows:

ES and BX - Address of the array (segment and offset)
CX - Length of the array
AL - Value to be searched for.

The result of the search is returned via AL register as follows

FFH - Value found in the table
00H - Value not found in the table.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	4B		DECW BX	;Adjust BX for initial entry into loop
2001	43	SR	INCW BX	;Update array pointer
2002	26		ES	
2003	38 07		CMPB [BX] AL	;Compare array element with the value in AL
2005	E0 FA		LOOPNE SR	;Continue search if CX not equal to 0 and array element not equal to AL.
2007	74 03		JE 200C	;If value found return true
2009	B0 00		MOVB AL #00	;Value not found return 0 in AL
200B	CC		INT 3	;Breakpoint to return to monitor.
200C	B0 FF	RT	MOVB AL #0FF	;Value found Return FF in AL
200E	CC		INT 3	;Return to monitor

1. Enter the above program from locations 2000H-200EH.
2. Now set up a byte array as follows from locations 2200H - 220FH

LOCATION	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2200	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45

3. Now set the values of the following registers as follows ES = 0220H ;BX=0H ;CX=10H ;AL=2FH.
4. Execute the program starting at location 2000H.
5. After the program is executed, control returns to monitor. Now examine AL register. Its contents should be 00 indicating that the value was not found in the table.
6. Repeat the experiment with the same values of ES, BX and CX but with AL=40. After the program is executed AL will have FF indicating that the value was found in the table.
7. Repeat the experiment with some other values in AL and verify that the program works properly.

8.3 USE OF MONITOR ROUTINES

As noted in chapter 6 several useful routines from the monitor programs are accessible to the user. All these user accessible routines are written as FAR (inter-segment) procedures. Appropriate use of these routines can lead to considerable simplification in the development of application software. The following examples illustrate the use of monitor routines in user programs.

Example 1: This example assumes that ESA 86/88-2 is operating in the keyboard mode.

The following program waits for a key to be pressed. Then the value of the key code is displayed in the data field with zero prompts. It then waits for another key pressing.

Note that the program is written as an endless loop. Hence, to recover from this program, you must press the RESET key.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	9A 1C 0B 00 FF	A1	CALLS 0B1C 0FF00	;Read a character from the keyboard
2005	A0 5A 00		MOVB AL 005A	;Set up parameters for output
2008	B4 00		MOVB AH #00	;No prompts
200A	9A 0A 0B 00 FF		CALLS 0B0A 0FF00	;Display key code
200F	EB EF		JMP A1	;Repeat

Load this program from 0:2000H and execute it using the GO command. Now, if you press any key (other than RESET, INTR keys), the corresponding code is displayed in the data field.

Example 2: This example also assumes that ESA 86/88-2 is operating in the keyboard mode.

The following program first blanks the data field with four prompts, displays the message `FIRE' in address field with zero prompts. After a delay, the address field is blanked with one prompt and the message `HELP' appears in the data field with zero prompts. The cycle is repeated for ever.

Note that the following program is written as an endless loop. Hence, to recover from this program, you must press the RESET key.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	B0 04	AG	MOVB AL #04	;With 4 prompts
2002	9A 00 0B 00 FF		CALLS 0B00 0FF00	;Blank data field
2007	B8 00 00		MOVW AX #0000	;Display `FIRE' in address field
200A	8E C0		MOVW ES AX	
200C	BB 40 20		MOVW BX #2040	;Starting address of array
200F	B0 01		MOVB AL #01	;Display address field
2011	B4 00		MOVB AH #00	;Zero prompts
2013	9A 31 0B 00 FF		CALLS 0B31 0FF00	
2018	E8 1D 00		CALL 2038	;Delay
201B	B0 01		MOVB AL #01	;With 1 prompt,
201D	9A 05 0B 00 FF		CALLS 0B05 0FF00	;Blank address field
2022	B8 00 00		MOVW AX #0000	
2025	8E C0		MOVW ES AX	
2027	BB 44 20		MOVW BX #2044	;Address of message `HELP'
202A	B0 00		MOVB AL #00	;Data field selected
202C	B4 00		MOVB AH #00	;Zero prompts
202E	9A 31 0B 00 FF		CALLS 0B31 0FF0	
2033	E8 02 00		CALL 2038	;Delay
2036	EB C8		JMP 2000	;Repeat
2038	B9 FF FF	DY	MOVW CX #0FFFF	
203B	E0 FE		LOOPNZ 203B	
203D	C3		RET	
2040	71 30	M1	DB 71 30	
2042	50 79		DB 50 79	
2044	76 79		DB 76 79	
2046	38 73		DB 38 73	

Load the above program in locations 0:2000H to 0:203DH and set up the arrays of messages to be displayed in locations 0:2040H to 0:2047H. Execute the program using GO command.

Example 3: This example also assumes that ESA 86/88-2 is operating in the keyboard mode.

The following program prompts for an address entry from the user and displays the word at the specified address and then halts. To recover, you must press the RESET key.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	B0 01		MOVB AL #01	;With one prompt
2002	9A 05 0B 00 FF		CALLS 0B05 0FF00	;Blank address field
2007	9A 1C 0B 00 FF		CALLS 0B1C 0FF00	;Get the first character into location 0:005AH
200C	B8 00 00		MOVW AX #0000	
200F	8E C0		MOVW ES AX	
2011	8D 1E 40 20		LEA BX 2040	;Pointer to hold the ;Received address value

2015	B8 00 00	MOVW AX #0000	;Default segment value
2018	B1 01	MOVB CL #01	;1 prompt
201A	9A 29 0B 00 FF	CALLS 0B29 0FF00	
201F	C4 1E 40 20	LES BX 2040	
2023	26	ES	
2024	8B 07	MOVW AX [BX]	
2026	B7 00	MOVB BH #00	;Data field
2028	B3 00	MOVB BL #00	;No prompts
202A	B1 00	MOVB CL #00	;No leading zero blanking
202C	9A 12 0B 00 FF	CALLS 0B12 0FF00	;Output the word
2031	F4	HLT	

This program uses four locations from 0:2040H as pointer to hold the received address value.

Example 4: This example assumes that ESA 86/88-2 is operating in the SERIAL mode.

The following program waits for a character from the user. After receiving the character, it checks if it is a valid hex character (`0' to `9' and `A' to `F'). If it is not a valid hex character, it displays a `?' and on a new line, the message, Invalid Hex. If the received character is valid hex character, it is converted to its binary equivalent and displayed on a new line along with the message valid hex value= '. Now the program waits for another character.

Note that this program is written as an endless loop. Hence, to recover from this program, you must press the RESET key.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	9A 70 1B 00 FE	AG	CALLS 1B70 0FE00	;Get a character
2005	88 C4		MOVB AH AL	; Preparatory to
2007	50		PUSH AX	Check if valid
2008	9A 77 1B 00 FE		CALLS 1B77 0FE00	;hex.
200D	3C FF		CMPB AL #0FF	;Valid hex?
200F	58		POP AX	;Restore ASCII input
2010	74 1B		JZ 202D	;Yes valid hex, continue
2012	B0 3F		MOVB AL #3F	NO
2014	9A 50 1B 00 FE		CALLS 1B50 0FE00	;Output a "?"
2019	9A 5B 1B 00 FE		CALLS 1B5B 0FE00	;Go to newline
201E	B8 00 00		MOVW AX #0000	;Output the
2021	8E C0		MOVW ES AX	;message
2023	B8 48 20		MOVW AX #2048	
2026	9A 55 1B 00 FE		CALLS 1B55 0FE00	
202B	EB D3		JMP AG	
202D	9A 80 1B 00 FE		CALLS 1B80 0FE00	;Convert to binary value
2032	50		PUSH AX	;Save the value
2033	B8 00 00		MOVW AX #0000	;Output the
2036	8E C0		MOVW ES AX	;message on a
2038	B8 54 20		MOVW AX #2054	;newline
203B	9A 55 1B 00 FE		CALLS 1B55 0FE00	
2040	58		POP AX	;Restore the value
2041	9A 64 1B 00 FE		CALLS 1B64 0FE00	;And output it

```
2046    EB B8                   JMP AG
2048    49 6E           M1      DB 49 6E
204A    76 61                   DB 76 61
204C    6C 69                   DB 6C 69
204E    64 20                   DB 64 20
2050    48 65                   DB 48 65
2052    78 00                   DB 78 00
2054    0D 0A           M2      DB 0D 0A
2056    56 61                   DB 56 61
2058    6C 69                   DB 6C 69
205A    64 20                   DB 64 20
205C    48 65                   DB 48 65
205E    78 2E                   DB 78 2E
2060    56 61                   DB 56 61
2062    6C 75                   DB 6C 75
2064    65 3D                   DB 65 3D
2066    20 00                   DB 20 00
```

Load the above program and message data into locations 0:2000 through 2067. For the convenience of the user, the contents are listed below.

LOCATION	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2000	**9A**	**70**	**1B**	**00**	**FE**	**88**	**C4**	**50**	**9A**	**77**	**1B**	**00**	**FE**	**3C**	**FF**	**58**
2010	**74**	**1B**	**B0**	**3F**	**9A**	**50**	**1B**	**00**	**FE**	**9A**	**5B**	**1B**	**00**	**FE**	**B8**	**00**
2020	**00**	**8E**	**C0**	**B8**	**48**	**20**	**9A**	**55**	**1B**	**00**	**FE**	**EB**	**D3**	**9A**	**80**	**1B**
2030	**00**	**FE**	**50**	**B8**	**00**	**00**	**8E**	**C0**	**B8**	**54**	**20**	**9A**	**55**	**1B**	**00**	**FE**
2040	**58**	**9A**	**64**	**1B**	**00**	**FE**	**EB**	**B8**	**49**	**6E**	**76**	**61**	**6C**	**69**	**64**	**20**
2050	**48**	**65**	**78**	**00**	**0D**	**0A**	**56**	**61**	**6C**	**69**	**64**	**20**	**48**	**65**	**78**	**2E**
2060	**56**	**61**	**6C**	**75**	**65**	**3D**	**20**	**00**								

1) Execute the above program from 0:2000.
2) Now enter a non-hex character (say Z). Observe the message to be: Invalid Hex.
3) Now enter a valid hex character (say F). Observe the display to be: valid hex value = 0F.
4) Try with a few other characters to ensure that the program is behaving as expected.

You can press the RESET key to recover from the program.

Example 5: This example also assumes that ESA 86/88-2 is operating in SERIAL mode.

The following program reads an address value from the user and displays the word at the specified address. Note that address is in the form of segment value: offset value. The address must be terminated by a comma or carriage return. If user does not specify a segment value, it uses the default value of 0H. After displaying the contents, the program control returns to the monitor.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	B8 00 00		MOVW AX #0000	;Set up required
2003	8E C0		MOVW ES AX	parameters and
2005	8D 1E 40 20		LEA BX 2040	CALL the required
2009	B8 00 00		MOVW AX #0000	monitor routine
200C	9A 90 1B 00 FE		CALLS 1B90 0FE00	
2011	C4 1E 40 20		LES BX 2040	
2015	26		ES	;Get the data and output it
2016	8B 07		MOVW AX [BX]	
2018	9A 69 1B 00 FE		CALLS 1B69 0FE00	
201D	CC		INT 3	;Return to monitor

The above program uses a pointer consisting of four locations from 0:2040 to hold the address value entered by the user.

Load the program from 0:2000 to 201D and execute it. Now specify a desired address (say 0:2000) and observe the contents of that location displayed (with the address of 0:2000, it must obviously be 00B8!).

Try with few other addresses and observe the program behavior.

Example 6: This example also assumes that ESA 86/88-2 is operating in the SERIAL mode.

The following program plays a simple guessing game with the user. It sets up a key value in a memory location and waits for a word input (a guess) from the user. If the user entered value is equal to the key value, it displays the message 'Correct Guess!!!' and returns to monitor. If the guess is less than or more than the key value, it displays appropriate message and again prompts for a guess.

The game can be terminated before guessing correctly by entering an ESC character when the system prompts for a guess. It will then display the key value and returns control to the monitor.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	B8 00 00	ST	MOVW AX #0000	
2003	8E C0		MOVW ES AX	
2005	B8 5D 20		MOVW AX #205D	
2008	9A 55 1B 00 FE		CALLS 1B55 0FE00	
200D	9A 70 1B 00 FE		CALLS 1B70 0FE00	;Get a character
2012	3C 1B		CMPB AL #1B	;Is it escape
2014	75 16		JNE 202C	;No
2016	B8 00 00		MOVW AX #0000	;Yes display key &
2019	8E C0		MOVW ES,AX	
201B	B8 6D 20		MOVW AX,#206D	
201E	9A 55 1B 00 FE		CALLS 1B55 0FE00	
2023	A1 00 01		MOVW AX 0100	;Get key and
2026	9A 69 1B 00 FE		CALLS 1B69 0FE00	display it
202B	CC		INT 3	

```
202C  9A 88 1B 00 FE   SE     CALLS 1B88 0FE00
2031  3B 06 00 01             CMPW AX, 0100
2035  77 10                   JNBE 2047
2037  72 13                   JB 204C
2039  B8 00 00                MOVW AX #0000
203C  8E C0                   MOVW ES AX
203E  B8 75 20                MOVW AX #2075
2041  9A 55 1B 00 FE          CALLS 1B55 0FE00
2046  CC                      INT 3
2047  B8 89 20                MOVW AX #2089
204A  EB 03                   JMP 204F
204C  B8 9D 20                MOVW AX #209D
204F  50                      PUSH AX
2050  B8 00 00                MOVW AX #0000
2053  8E C0                   MOVW ES AX
2055  58                      POP AX
2056  9A 55 1B 00 FE          CALLS 1B55 0FE00
205B  EB A3                   JMP ST
205D  0D 0A            M1     DB 0D 0A
205F  59 6F                   DB 59 6F
2061  75 72                   DB 75 72
2063  20 67                   DB 20 67
2065  75 65                   DB 75 65
2067  73 73                   DB 73 73
2069  20 3D                   DB 20 3D
206B  20 00                   DB 20 00
206D  0D 0A            M2     DB OD OA
206F  4B 65                   DB 4B 65
2071  79 3D                   DB 79 3D
2073  20 00                   DB 20 00
2075  0D 0A            M3     DB 0D 0A
2077  43 6F                   DB 43 6F
2079  72 72                   DB 72 72
207B  65 63                   DB 65 63
207D  74 20                   DB 74 20
207F  67 75                   DB 67 75
2081  65 73                   DB 65 73
2083  73 21                   DB 73 21
2085  21 21                   DB 21 21
2087  21 00                   DB 21 00
2089  0D 0A            M4     DB 0D 0A
208B  54 72                   DB 54 72
208D  79 20                   DB 79 20
208F  73 6D                   DB 73 6D
2091  61 6C                   DB 61 6C
2093  6C 65                   DB 6C 65
2095  72 20                   DB 72 20
2097  76 61                   DB 76 61
2099  6C 75                   DB 6C 75
209B  65 00                   DB 65 00
209D  0D 0A            M5     DB 0D 0A
```

```
209F    54 72          DB 54 72
20A1    79 20          DB 79 20
20A3    6C 61          DB 6C 61
20A5    72 67          DB 72 67
20A7    65 72          DB 65 72
20A9    20 76          DB 20 76
20AB    61 6C          DB 61 6C
20AD    75 65          DB 75 65
20AF    00             DB 00
```

For the convenience of the user, the contents are listed below

LOCATION	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2000	**B8**	**00**	**00**	**8E**	**C0**	**B8**	**5D**	**20**	**9A**	**55**	**1B**	**00**	**FE**	**9A**	**70**	**1B**
2010	**00**	**FE**	**3C**	**1B**	**75**	**16**	**B8**	**00**	**00**	**8E**	**C0**	**B8**	**6D**	**20**	**9A**	**55**
2020	**1B**	**00**	**FE**	**A1**	**00**	**01**	**9A**	**69**	**1B**	**00**	**FE**	**CC**	**9A**	**88**	**1B**	**00**
2030	**FE**	**3B**	**06**	**00**	**01**	**77**	**10**	**72**	**13**	**B8**	**00**	**00**	**8E**	**C0**	**B8**	**75**
2040	**20**	**9A**	**55**	**1B**	**00**	**FE**	**CC**	**B8**	**89**	**20**	**EB**	**03**	**B8**	**9D**	**20**	**50**
2050	**B8**	**00**	**00**	**8E**	**C0**	**58**	**9A**	**55**	**1B**	**00**	**FE**	**EB**	**A3**	**0D**	**0A**	**59**
2060	**6F**	**75**	**72**	**20**	**67**	**75**	**65**	**73**	**73**	**20**	**3D**	**20**	**00**	**0D**	**0A**	**4B**
2070	**65**	**79**	**3D**	**20**	**00**	**0D**	**0A**	**43**	**6F**	**72**	**72**	**65**	**63**	**74**	**20**	**67**
2080	**75**	**65**	**73**	**73**	**21**	**21**	**21**	**21**	**00**	**0D**	**0A**	**54**	**72**	**79**	**20**	**73**
2090	**6D**	**61**	**6C**	**6C**	**65**	**72**	**20**	**76**	**61**	**6C**	**75**	**65**	**00**	**0D**	**0A**	**54**
20A0	**72**	**79**	**20**	**6C**	**61**	**72**	**67**	**65**	**72**	**20**	**76**	**61**	**6C**	**75**	**65**	**00**

Load the above program and message data from 0:2000 and execute it and have fun!

8.4 USE OF 8087 CO-PROCESSOR

ESA 86/88-2 provides a socket for installing the optional Numeric Data processor-8087. As ESA 86/88-2 is based on 8086/8088 operating in the maximum mode, the support for 8087 is direct. You have to just install the device in the appropriate socket; no other changes are required.

The following two examples illustrate the use of 8087. After installing the 8087, the following examples can be executed either from the keyboard monitor or from the serial monitor.

Example:1

The following program assumes that two 32-bit integer data values, a and b, are given starting from the locations 3000H and 3004H respectively. It then computes a result C=square root of (a2+b2) and stores the result as a 32-bit integer starting at location 3008H. (code segment value is assumed as 0000)

NOTE: On-board Assembler, ESAM-86, does not support NDP instructions. So these programs must be entered as hexadecimal values.

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	BB 00 30		MOV BX,#3000H	;Point to `a'
2003	DB 07		FLD (BX)	;Load `a'
2005	9B		FWAIT	
2006	DA 0F		FMUL (BX)	;Compute a2
2008	9B		FWAIT	
2009	DD D9		FST ST (1)	;Save a2 in ST (1)
200B	9B		FWAIT	
200C	81 D3 04 00		ADD BX,#0004	;Point to `b'
2010	DB 07		FLD (BX)	;Load `b'
2012	9B		FWAIT	
2013	DA 0F		FMUL (BX)	;Compute b2
2015	9B		FWAIT	
2016	D8 C1		FADD ST(1)	;ST (0)=a2+b2
2018	9B		FWAIT	
2019	D9 FA		FSQRT	;ST(0)=Sqrt(a2+b2)
201B	9B		FWAIT	
201C	81 D3 04 00		ADD BX,#0004	;Point to location for C
2020	DB 17		FST (BX)	;Store the result in c ;as a 32-bit integer
2022	9B		FWAIT	
2023	CC		INT 3	Return to monitor

1. Load the above program into memory.
2. Set up data values a and b as follows.

   3000 : 03, 00, 00, 00
   3004 : 04, 00, 00, 00
3. Execute the program and observe the result. It should be as follows:

   3008 : 05, 00, 00, 00

4. You can try with different data values to observe the program behaviour

Example 2:

The following program calculates Sin(Z) where Z is in degrees and $0 < Z < 90$. Note that the values of 0 and 90 are not allowed. They must be handled separately (the reason for which will become clear shortly).

Sin(Z) is calculated using the Tangent function (FPTAN) of 8087 as follows:

Suppose tan (Z/2) = Y/X. Then
Sin (Z) $= 2XY/(X^2+Y^2)$

The FPTAN function of 8087 requires the argument, in radians, on the top of the stack and returns the result as Y/X where X is the top of stack and Y is the next element. Further the argument must satisfy

0< argument <PI/4. This is the reason we restricted Z to satisfy 0<Z<90. This restriction can be eliminated, at the cost of more computation as follows:

Let Z be unrestricted. Then, use the FPREM function of 8087 to reduce the argument to the required range and use the relation Sin (PI+Z) = -Sin (Z)

In the program given below, the argument Z is first divided by 2 (preparatory to using the relation described above), then converted into radians and then the tangent calculated. The resulting values X and Y are used to calculate sin (Z).

The input to the program, Z (in degrees), must be set up in the register AL, as a hexadecimal number. The output, in packed BCD form is available in the register AX. (A decimal point is to be assumed before the first BCD digit).

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	B4 00		MOV AH, #00	
2002	D1 E8		SHR AX,1	;Z=Angle = Angle/2
2004	A3 5A 20		MOVW Arg1, AX	;Preparatory for
2007	9B DF 06 5A 20		FLDI Arg1	;loading into NDP.
200C	C7 06 5C 20 B4 00		MOVW Arg2, #00B4	;(Arg2) = 180
2012	9B DF 06 5C 20		FLDI Arg2	;Insert a FWAIT and load Arg2.
2017	9B DE F9		FDIVRP	;Wait and divide (ST) = Z/180
201A	9B D9 EB		FLDPI	;(ST)=PI
201D	9B DE C9		FMULRP	;(ST)=PI*Z/80 = angle radians
2020	9B D9 F2		FPTAN	;Compute tan as Y/X
2023	9B D9 C0		FLD ST(0)	;Duplicate X on stack Top
2026	9B D8 C9		FMUL ST,ST(1)	;ST=X*X
2029	9B D9 C2		FLD ST (2)	;ST=Y
202C	9B D8 CB		FMUL ST,ST(3)	;ST=Y*Y
202F	9B DE C1		FADDP ST(1),ST	;Y*Y is popped off & ;ST= X*X + Y*Y
2032	9B DE F9		FDIVRP	;ST= X/(X*X+Y*Y)
2035	9B DE C9		FMULRP	;ST= X*Y/CX*X+Y*Y)
2038	9B D9 C0		FLD ST(0)	;Duplicate the value on stack top
203B	9B DE C1		FADDP ST(1),ST	;ST=2*X*Y[X*X-Y*Y)
203E	C7 06 5E 20 10 27		MOVW Arg3, #2710H	
2044	9B DF 06 5E 20		FLDI Arg3	
2049	9B DE C9		FMURP	;Result=10,000 *s in(Z)
204C	9B D9 FC		FRNDINT	; Round to integer
204F	9B DF 36 60 20		FSTP Arg 4	;and store in BCD Form.
2054	9B		FWAIT	;and get the result into
2055	A1 60 20		MOV AX, Arg 4	;AX and return to
2058	CC		INT 3	;monitor

For the convenience of the user, the contents are listed below

LOCATION	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2000	B4	00	D1	E8	A3	5A	20	9B	DF	06	5A	20	C7	06	5C	20
2010	B4	00	9B	DF	06	5C	20	9B	DE	F9	9B	D9	EB	9B	DE	C9
2020	9B	D9	F2	9B	D9	C0	9B	D8	C9	9B	D9	C2	9B	D8	CB	9B
2030	DE	C1	9B	DE	F9	9B	DE	C9	9B	D9	C0	9B	DE	C1	C7	06
2040	5E	20	10	27	9B	DF	06	5E	20	9B	DE	C9	9B	D9	FC	9B
2050	DF	36	60	20	9B	A1	60	20	CC							

Note that 16 locations 205A to 206A (ARG1,ARG2,ARG3 and ARG4) are used for storing the constants and results.

1. Load the above program into the memory.
2. Set up the input parameter in the register AL (for example, to calculate Sin 60, (AL) = 3C H)
3. Execute the program and observe the contents of AX (with input as (AL) = 3C, (AX) will now be 8660 so Sin 60 = 0.8660)
4. Repeat the program with different input data and observe the output as shown below.

Input (AL)	Output (AX)	Calculated function
1E	5000	sin 30
2E	7193	sin 46
10	2756	sin 16
56	9976	sin 86

Exercises:

1. Observe that above program divides the input by 2 by shifting AX. However this will loose the least significant bit if the original value is odd. For example, with (AL)= 2D shifted AX will contain 16 (=22 decimal) and thus the argument used will be 16H. This leads to the calculation of sin (2x22) i.e sin 44 as opposed to the desired sin 45! (you can run the program with (AL)= 2D and observe that (AX)=6946, while sin 45=.7071!!)

   Thus an accurate version should load the original angle value into 8087 and the division by 2 must be performed by the 8087. Change the above program to implement this modification.

2. The above program produces a result rounded to 4 digits (Notice the multiplication of result by 10000 and rounding to integer value). How can you improve the accuracy?

3. Based on the approach presented here, develop a program to calculate cos Z where $0 < Z < 90$.

4. Using the FPREM function of 8087, calculate sin Z for any value of Z.

8.5 USE OF PROGRAMMABLE INTERRUPT CONTROLLER-8259A.

ESA 86/88-2 provides a versatile and powerful interrupt system design capability to the user. There is an 8259A on-board, which can accept 8 interrupt requests either from on-board sources or from off-board sources. This desired configuration can be set up via jumper options as described in section 2.1.6.
The following illustrative example demonstrates the use of 8259A in a simple application context.

Example:

In this example, the source of interrupt is selected as Timer 1 of 8253-A and it is connected to INT5 of 8259. For this purpose, you must establish the following jumper :

JP3 - 2-3 Timer 1 of 8253 to IR5 of 8259A

To utilize Timer 1 of 8253A, its gate (ie Gate1) must be tied to Vcc and its clock (clock 1) must be connected to appropriate clock source. In this example, we will use the PCLK (peripheral clock) for clock 1. For these purposes, you must establish the following connections.

Pin 9 of connector J1 to Vcc. ; Gate 1 to Vcc
Pin 41 of connector J1 to pin 14 of connector J1 ; Clock 1 to PCLK

We will program 8259A with base value for Type as 72(decimal). As interrupt from Timer 1 occurs on IR5, 8259A will send out a type code of 72+5=77 (decimal) Thus a pointer for interrupt service routine must be stored in location 77X4 (decimal) i.e in locations 0134H, 0135H, 0136H, 0137H. Assume we are going to write our service routine from 0:2050H. Then we must set up this pointer value. As the locations 0:134 to 0:137 are used by the on-board assembler, the pointer must be set up in the program itself.

The main program now follows:

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2000	BC 00 30		MOVW SP,#3000	;Allow more stack area
2003	C7 06 34 01 50 20		MOVW 0134,#2050	;Offset of interrupt ;service routine
2009	C7 06 36 01 00 00		MOVW 0136,#0000	;Segment of interrupt ;service routine
200F	BA F4 FF		MOVW DX,#0FFF4	;Address for ICW1
2012	B0 13		MOVB AL,#13	;Edge - triggered, ;single 8259, ICW4 ;needed.
2014	EE		OUTB DX	
2015	BA F6 FF		MOVW DX,#0FFF6	;Address for ICW2, ;ICW4
2018	B0 48		MOVB AL,#48	;Base value of ;Type is 72(decimal

201A	EE		OUTB DX	
201B	B0 03		MOVB AL,#03	;Auto EOI, 86/88 ;mode for 8259.
201D	EE		OUTB DX	
201E	B0 DF		MOVB AL,#0DF	;Mask off all ;levles execpt IR5
2020	EE		OUTB DX	
2021	BA FF FF		MOVW DX,#0FFFF	
2024	B0 70		MOVB AL,#70	;Counter 1 in Mode ;0 (interrupt on Terminal ; count mode)
2026	EE		OUTB DX	
2027	BA FB FF		MOVW DX,#0FFFB	
202A	B0 00		MOVB AL,#00	;Low byte of count
202C	EE		OUTB DX	
202D	C6 06 68 20 00		MOVB 2068,#00	;Set up a byte to ;Zero, as a flag.
2032	EE		OUTB DX	;High byte of count
2033	FB		STI	;Enable interrupts
2034	80 3E 68 20 FF	W1	CMPB 2068,#0FF	;Has flag changed? ;(because of ;interrupt)
2039	75 F9		JNZ W1	;No, wait for interrupt
203B	CC		INT 3	;Interrupt has occurred ;Return to Monitor.

Now, we must write a service routine at 2050H. Let us write a routine which will display the message "Interrupt from Timer 1".

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2050	B8 00 00		MOVW AX,#0000	;Set up parameters
2053	8E C0		MOVW ES,AX	;for calling monitor
2055	B8 70 20		MOVW AX,#2070	;routine to display
2058	9A 55 1B 00 FE		CALLS 1B55 0FE00	;the message
205D	C6 06 68 20 FF		MOVB 2068,#0FF	;Change the flag
2062	CF		RTI	;Return from interrupt

LOC	CONTENTS	LABEL	INSTRUCTION	COMMENTS
2070	0D 0A 49 6E		DB 0DH,0AH, 'Interrupt from	
2074	74 65 72 72		Timer 1', 0DH,0AH,00	
2078	75 70 74 20			
207C	66 72 6F 6D			
2080	20 54 69 6D			
2084	65 72 20 31			
2088	0D 0A 00			

For the convenience of the user, the codes are listed below.

LOCATION	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2000	BC	00	30	C7	06	34	01	50	20	C7	06	36	01	00	00	BA
2010	F4	FF	B0	13	EE	BA	F6	FF	B0	48	EE	B0	03	EE	B0	DF
2020	EE	BA	FF	FF	B0	70	EE	BA	FB	FF	B0	00	EE	C6	06	68
2030	20	00	EE	FB	80	3E	68	20	FF	75	F9	CC				
2040																
2050	B8	00	00	8E	C0	B8	70	20	9A	55	1B	00	FE	C6	06	68
2060	20	FF	CF													
2070	0D	0A	49	6E	74	65	72	72	75	70	74	20	66	72	6F	6D
2080	20	54	69	6D	65	72	20	31	0D	0A	00					

Now execute the program at 0:2000. You should get the message `Interrupt from Timer1' and then control should return to the monitor.

As already noted, locations 0:0134 to 0:0137 are used by the on-board assembler. Hence when the monitor regains control, it finds that these locations are altered and issues the message "Assembler corrupted ! Being restarted !!" before issuing the sign-on message. Thus user can ignore this message when using 8259A.

Exercise: Use Timer 2 of 8253-5 as a second interrupt source and program 8259-A for a variety of interrupt operations for experimenting with priorities etc.

CHAPTER 9

AUDIO TAPE INTERFACE

9.1 INTRODUCTION

The audio tape interface for ESA 86/88-2 is an optional facility and it consists of the necessary hardware and software to allow the user to store data on and read data from any commercial Audio Cassette Recorder. The data is stored and retrieved as named files.

The interface can be operated either from the keyboard Monitor or from the Serial Monitor.

9.2 INSTALLATION

The interface unit includes a Tone-out LED, two sockets marked as MIC and EAR and a four-lead connector (one lead for data out, one for data in and two for power supply). The connector is used for interfacing the unit to ESA 86/88-2 via the connector J4 of ESA 86/88-2. (Please refer to the ESA 86/88-2 Component placement diagram for position details).

❖ ***IMPORTANT***

The power supply for the interface unit is derived from the ESA 86/88-2 via the connector. Hence no external power supply connections are required.

The MIC and EAR sockets provided on the interface unit are for connecting, respectively to the MIC and EAR sockets of the Tape Recorder. An interconnection cable is also supplied for this purpose.

❖ ***CAUTION***

Connect only one socket MIC or EAR at any time to the Recorder. Connecting both the sockets permanently to the tape recorder may short circuit both the signal lines if the recorder common point connection is different from the one assumed by ESA 86/88-2 Interface.

9.3 OPERATION FROM SERIAL MONITOR

The Serial Monitor of ESA 86/88-2 supports two commands for using the audio tape interface. These commands are:

T : Transfer from memory to Tape.
L : Load into memory from tape.

9.3.1 T command: Transfer from memory to Tape.

This command is used to transfer information from memory on to the tape. The information is saved as a named file.

To use this command, type T when ESA 86/88-2 prompts for a command (with the dot prompt).

The system will now prompt for a file name as shown below:

File Name:

The file name consists of a maximum of 4 characters, each of which must be a valid hexadecimal digit. Some examples are shown below:

Valid File names	**Invalid File names**
14B3	134G (G is not a hexadecimal digit)
A214	FA3Z (Z is not a hexadecimal digit)
A5A5	FF.1 (. is not a hexadecimal digit)
F123	22TT (T is not a hexadecimal digit)

Enter the desired file name and terminate it with a carriage Return.

Now the system will prompt for the starting address of the memory block to be saved on the tape:

Buffer–start =

Enter the starting address in segment: offset form and terminate the address entry with a Carriage Return. (If no segment value is entered, the current value of the user's code segment register is used).

Now the system will prompt for the ending address of the memory block:

Buffer-end =

Note that Buffer-end address consists of only offset information. No segment value is allowed with the end-address. (Thus at a time, a maximum of 64K bytes can be transferred to the tape.) Further note that this value must be greater than or equal to the offset of the starting address.

Enter the desired end address and terminate it with a Carriage Return.

Now the system will display the message

Press any key after setting up the Recorder.

- Ensure that the Recorder is connected to the interface via the MIC jack only.
- Set up the Recorder in record mode.

Now type any character (except the ESC key which will return control to the monitor).

The system will now display the message :

Writing to tape ....

It will begin transferring the data from the specified memory block onto the tape. After the data, a checksum byte is also written onto the tape. During this period the TONEOUT LED will be ON. After completing the transfer, control will return to the Monitor (which will then issue the command prompt).

❖ ***IMPORTANT***

Data will be sent out once the user responds to the message: Press any key after setting up the Recorder. If recorder is really not ready at this stage, the data is still sent out and this will not get recorded on the tape!

EXAMPLE

Store the contents of locations 0:2000 to 207F on tape with a file name 123F.

The required interaction is shown below: (Note that the information to be entered by the user is underlined only for the sake of clarity).

* T

File Name = 123F
Buffer start = 0: 2000
Buffer End = 207F

Press any key after setting up the Recorder
Writing to tape .....

9.3.2 L command : Load from Tape

This command is used to read information from the tape into memory.

To use this command, type L when prompted for command entry by the Serial Monitor.

Now the system will prompt for the filename as shown below:

File Name =

Now enter the file name. (The rules regarding the filename are same as the ones explained in T command). Note that the addresses of the memory block are already stored on the tape under the specified filename. So user need not enter them again. Terminate the filename entry with the Carriage Return.

Now the system will display the message:

Reading from tape .....

- Ensure that the Recorder is connected to the interface via the EAR jack only.
- Set the Recorder in the PLAY mode.

As the system searches for the desired file, the data read from the tape is echoed onto the TONE OUT LED. Further, every file name read by the system during the search is displayed on a new line as shown below. (xxxx will be actual value read)

File Name = xxxx

When the specified file is found, the system will display the message.

Found the Specified file. Loading .....

As already noted, the start and end addresses of the block are read from the tape itself. Further, a checksum byte recorded on the tape during Tape write operation, is also read. The checksum byte is compared with the checksum calculated from the actual data read from the tape.

If the file is loaded without any errors, the following message is displayed.

File Loaded

Now the control returns to the Monitor.

If any errors occur during the read process, the system will display the message:

Read Error

Errors can occur because of excessive tape speed variations, noise problems etc. After displaying the error message, the system will ask:

Retry ? (Y/N)

If you want to retry the loading process, type Y. Typing N will return control to the Monitor. If Y is typed, the system will display the message: **Reposition the Recorder and type any key**

Rewind the tape recorder beyond the expected starting position of the desired file and type any key (except the ESC key). The system will make another Read attempt.

Example

Read the contents of the file with filename of 123F (Assume the tape is positioned such that there are intervening files. User response is underlined for the sake of clarity only).

```
.L
File Name = 123F
Reading from tape .....
File Name = 5432
File Name = 3432
Found the specified file. Loading .....
File loaded
.
```

9.3.3 Response to Entry errors.

Whenever the user types a character which is invalid in the sequence, the system will display the message

What ???

and will repeat the prompt in response to which the error was detected.

9.4 OPERATION FROM KEYBOARD MONITOR

The keyboard Monitor of ESA 86/88-2 supports two command keys TAPE WR and TAPE RD for using the audio interface. The keys are used as:

TAPE WR : Tape Write (Transfer from memory to Tape)
TAPE RD : Tape Read (Load into memory from Tape).

9.4.1 TWR Command : Transfer from memory to Tape

This command is used to transfer information from memory onto the tape. The information is saved as a named file.

To use this command, press TAPE WR key when the monitor prompts for a command.

Now connect the microphone of the recorder to the MIC jack of the Tape interface Hardware and set the Recorder in record mode and enter the file parameters, from the hex keypad of ESA 86/88-2. The parameters are entered as follows:

Filename, NEXT starting address, NEXT Ending address EXEC

Filename = a string of upto 4 hexadecimal digits

Starting address = segment: offset

The starting address of the block of data to be transferred to the tape. Note that this address is in segment: Offset form.

Ending address = Offset

The ending address of the block of data to be transferred to the tape. Note that no segment value is permitted with the ending address.

Now data will be transferred on to the tape. After the data, a checksum byte is also written onto the tape. During the transfer, the display will be blank and the TONE OUT LED will be ON. After transferring the data, control returns to the MONITOR.

IMPORTANT

If the recorder is not ready and you press the EXEC key, data is still sent out and this data will not be recorded on tape. Hence ensure that the recorder is ready before pressing the EXEC key.

Example: To store the contents of location 2000 to 217F on tape with a file name 123F.

Key pressed	**Display**		**Comment**
	Address field	**Data field**	
RESET	- E S A	8 6	System Reset (with 8086 CPU)
TAPE WR	. . .		Tape write command
1	0 0 0 1.		Prompt for filename
2	0 0 1 2.		
3	0 1 2 3.		Filename = 123F
F	1 2 3 F.		
,(NEXT)	. .		Prompt for starting address
0	0 0 0 0.		
:	0 0 0.0.		
2	0 0 0 2.		
0	0 0 2 0.		
0	0 2 0 0.		Starting address=0:2000H
0	2 0 0 0.		Prompt for ending address.
,(NEXT)			
2	0 0 0 2.		
1	0 0 2 1.		
7	0 2 1 7.		
F	2 1 7 F.		Ending address=217FH Connect the microphone of the recorder to MIC of Interface Unit. Set the Recorder in record mode. Data is being recorded on the tape Transfer is completed. Control returns to Monitor.
EXEC			
(.)			

9.4.2 TAPE RD Command: Load into memory from Tape

This command is used to read information from the tape into the memory.

To use this command, press TAPE RD key when prompted for command entry by the keyboard monitor.

Now the system will prompt for a file name. Enter the filename of upto 4 hexadecimal digits terminated by the Period (.) key (i.e. EXEC key)

Connect the Earphone of the recorder to EAR jack of the Tape Interface Hardware. Set the recorder in PLAY mode.
The display will be blank. The data read from the tape is echoed on the TONEOUT LED. Every filename read by the system will be displayed for approximately 2 seconds. If the displayed file name is not the specified one, display is again blanked with a dot in the address field. When the specified

file is found, the entire display will be blank, with four dots in the address field. Data is transferred from the tape into the memory. As noted already, the starting and ending addresses of the memory block are read from the tape itself. The checksum recorded on the tape during tape write, is read and is compared with the checksum of the actual data transferred. If they do not match, or if the data transfer is not correct, the system displays error, control returns to the monitor which will display the command prompt dash.

In such case, rewind the tape, check for loose connections and re-execute the command. If the error was due to noise problems, it should be possible to read successfully during the second try.

EXAMPLE: To read the contents of the file with the filename of 123F.

Key pressed	**Display**		**Comment**
	Address field	**Data field**	
RESET	- E S A	8 6	System Reset (with 8086 CPU)
TAPE RD	.		Tape Read command
1	0 0 0 1.		
2	0 0 1 2.		
3	0 1 2 3.		
F	1 2 3 F.		Filename=123F
EXEC			Connect recorder (using earphone jack) to the EAR of the tape Interface Hardware.
(PLAY)			Press play of the recorder.
	.		Display is blank.. While system searches for filename.
	X X X X		First filename found displayed for 2 seconds.
	.		It is not the desired file. Again display is blanked with a dot; search continues. This process repeats until the required file is found.
	1 2 3 F		Display for 2 seconds. It is the desired file. The entire display is
	. . . .		blank with four dots in address field Data is transferred from tape into memory.
	-		Completion of data transfer successfully

9.4.3 Response to entry errors

If an invalid key is pressed at any time, `-Err' message is displayed and control returns to the monitor.

9.5 General Hints

The audio tape interface is fairly reliable. However, to improve the reliability of the operation, the following general hints may be useful.

- While recording and reading, ensure that volume and Tone controls are set sufficiently high.
- Use a good quality recorder. More important is a good quality cassette. Use a high - fidelity cassette for reliable operation.
- The interface can accommodate normal variation in the tape speed. Ensure that your recorder does not produce too significant changes in the tape speed.
- Avoid storing relatively large files. If the file is really large split into multiple small files. Though this scheme reduces the utilization of the tape space, it improves the chances of restoring the complete file correctly.

9.6 THEORY OF OPERATION

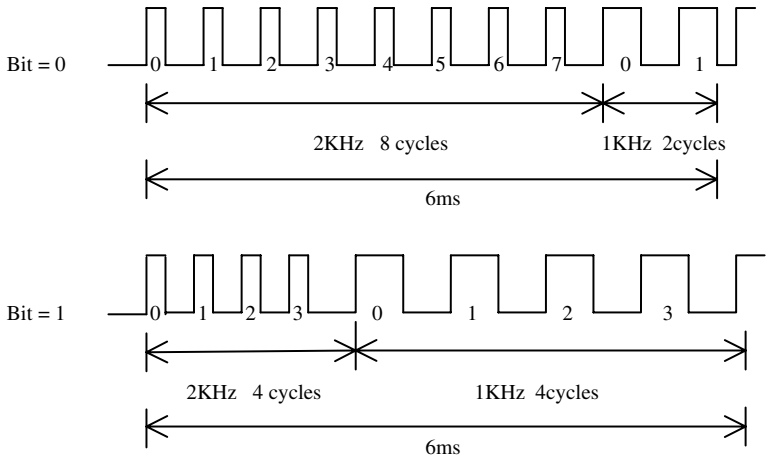
The program or data to be stored on the tape is sent out, in a serial fashion, via PC7 of the lower 8255 (with addresses FFE0, FFE2, FFE4 and FFE6). PC7 is connected to a passive filtering and attenuating circuit. The attenuated signal is fed to the MIC of the Recorder.

The file is read back from the EAR outlet of the recorder. The signal from the recorder is shaped and converted to TTL levels and is detected by the CPU via PA0 of the lower 8255.

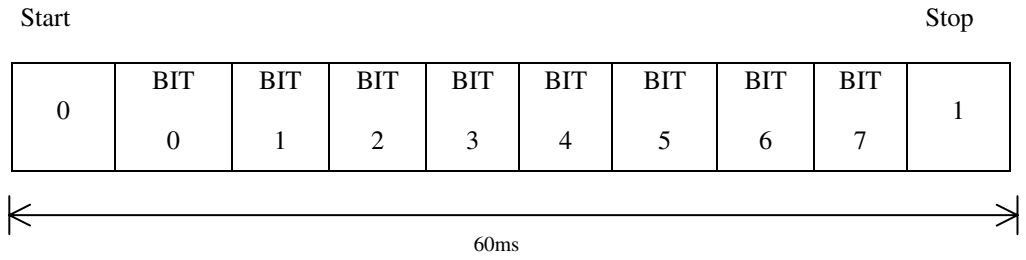
9.6.1 Data Format

Information is recorded on the Tape according to following formats

1) Bit Format: Both 0 and 1 are recorded as combination of some high frequency (2KHz) signals and some low frequency (1KHz) signals as shown below:



ii) Byte format: A byte is recorded as one start bit (a zero) 8 data bits and one stop bit. (a one).



iii) File format: Before recording the file, a lead synchronization signal of 1 KHz frequency is recorded for 4 seconds. Then a nine byte header is recorded. The file header consists of filename (2 Bytes), start address (4 bytes), end address (2 bytes) and checksum (1 byte). Following this, a mid synchronization signal is recorded. This consists of a 2KHz signal for 2 secs and 1KHz signal for 0.8 secs. Now the file data is recorded. After this, a tail synchronization signal of 2KHz frequency is recorded for 2 seconds. Thus the file format is:

9.6.2 Storage Capacity:

As is obvious from the above description of the Data formats, the exact amount of data that can be stored on a tape, depends upon the total number of files created. The larger the number, the smaller is the amount of data that can be stored (because of the space consumed by the larger number of synchronization signals). Assuming a typical number of 25 files for a standard C-90 tape of 90 minutes duration, we see that approximately 10 minutes (after allowing for about 15 seconds of inter-file gap during recording) may be taken up by synchronization signals. This allows us to record approximately 72 KB of data on one such cassette. However, if data is recorded in files of smaller size (say 128 or 256 bytes/file), the number of files stored will increase and total data storage capacity may fall to about 40K bytes.

CHAPTER 10

PROM PROGRAMMER SYSTEM

10.1 INTRODUCTION

ESA-86/88-2 PROM programmer system is a powerful and easy to use optional attachment for ESA - 86/88-2. This chapter describes the use of this PROM Programmer system.

The system permits the user to program, verify, blank check and read any of the popular EPROMs - 2716 through 27512. The system consists of a hardware module and the necessary software. The software can be invoked either from the keyboard monitor or from the serial monitor. A 28 pin ZIF socket is provided for placing the EPROMs. When a 24-pin EPROM is to be placed, it must be aligned with the bottom row i.e. top two rows of pins are to be left blank.

The system uses Intelligent Programming Algorithm whenever possible which reduces the programming time significantly.

The devices supported by the system and the type number to be entered by the user are listed below:

Device	**Type Number to be entered by the user**
2716	2716
2732	2732
2732A	732A
2764	2764
2764A (27C64)	764A
27128	0128
27128A (27C128)	128A
27256@12.5V	0256
27256@21V	2256
27512	0512

The device selection is totally software-controlled and no hardware changes or jumper settings are necessary for selecting any of the above listed devices.

10.2 INSTALLATION

1. Turn off power to ESA-86/88-2 trainer.
2. Attach the hardware module to ESA-86/88-2 over the connector J5 using the 26 core ribbon cable supplied with the module.
3. Connect the black, red, yellow and blue wires coming from the four pin polarized connector on the programmer module, to the corresponding coloured point of ESA power supply as shown below:

Colour of wire	Supply to be connected
RED	+ 5V
BLACK	GROUND
YELLOW	+12V
BLUE	+30V

If a separate supply is used for programmer module, ensure that its common is connected to the common of the power supply used for the trainer.

4. Turn On the power and press the RESET KEY; now the PROM programming system is ready for operation.

➢ ***CAUTION***

i. Spurious application of programming voltage to EPROM while ESA-86/88-2 is being switched ON / OFF may cause damage to the EPROM. To avoid this, insertion/removal of EPROM should be done only when ESA-86/88-2 is powered ON.

ii. Make sure that the selected type is correct for the EPROM inserted in ZIF socket.

iii. Avoid attempting to program already damaged devices.

➢ ***Warning:***

Failing to take any of the above precautions may lead to the damage of the device in use as well as the system.

10.3 Operation from Serial Monitor

To invoke the programmer system from the Serial Monitor, type P when prompted for a command. Now, the PROM programmer software gains control of the system and it will issue its own sign-on message as follows:

ESA 86/88 PROM PROGRAMMER V1.0

On the next line , a command prompt ('*') followed by a list of the commands is displayed as follows:

* (P/R/V/B/E)

The system is mostly menu driven. As shown above, the available commands are P,R,V,B and E which have the following meaning.

P - Program
R - Read
V - Verify
B - Blank check
E - Exit to the System Monitor.

Enter the appropriate command. If an invalid character is entered, the system displays the error message " what ????" and again prompts for a command .

Aborting a command:

Once a specific command is issued, further prompts will depend on the command itself. However, if the user enters ESC (in the sub-menu) whenever the system is looking for an entry from the user, the current operation is aborted and control returns to the programmer section menu.

10.3.1 P command:

This command is used to program a PROM.

This command requires the following four parameters:

Type	= PROM Type : should be one of the types listed above in section 10.1
Buffer Start	= Starting address of the source of data (It can be in `segment : offset form or only offset form. If only offset is specified, a default segment value is used which is equal to the value of the user's code segment register).
Buffer End	= Ending address of the source of data. It can be specified only as offset. No segment values are permitted.
PROM start	= Absolute starting address of the PROM (from where programming is to begin)

a) As soon as `P' is typed, the system will display the current values for all of the above four parameters. Then it asks if any change is required through a prompt as shown below:

Any change of the above parameters ?(Y/N)

If the displayed parameter values are all right type N (for No). On the other hand, if you wish to change any value, enter Y (for yes). Any other character produces error message "what ????" and the prompt reappears.

If user response to above prompt is Y, the system displays each parameter value and prompts for new value. User can enter a new value followed by Carriage Return or simply enter Carriage Return if the displayed value is not to be changed.

Note that the parameters must satisfy certain conditions as listed below.

i) PROM type can be only be one of the types listed in section 10.1.
ii) Both Buffer Start and Buffer End must be even addresses or odd addresses.

➢ ***IMPORTANT***

8086 memory space consists of an even bank and an odd bank. Thus a PROM is always programmed from alternate source bytes i.e. bytes from successive even addressed locations or bytes from successive odd addressed locations (strip even/strip odd) . Though 8088 memory space is linear, on ESA 86/88-2 it is organized as even/odd arrays !!!.

iii) Buffer end address can be only an offset (Segment is not allowed). The segment value is the one specified for the Buffer start address.

iv) Buffer end address must be greater than or equal to the offset value of the Buffer start address.

v) The PROM must have enough space to accommodate all the bytes specified by the Buffer start address and Buffer end address. In other words, the following relation must be satisfied.

PROM Start + [(Buffer end address-Buffer start address)/2]
<=Highest absolute address of the PROM.

For example, suppose PROM type is 2764. Then its highest absolute address is 1FFF H. Suppose the other parameters are as follows:

Buffer Start	=	FC00 : 0000
Buffer end	=	3FFE
PROM Start	=	100

Then 100 + (3FFE-0000)/2 = 20FF > 1FFF so this combination of parameters is invalid.

As user enters the parameter values, the above mentioned constraints are checked and if any of the constraints are violated, the error message `What ???' is displayed and user is again prompted for a value for the parameter which resulted in the detection of the error.

After optional modification of the parameter values by the user, the system checks the PROM for blank values (0FFH) in the required zone. If the PROM is not blank, the following prompt appears:

PROM is not blank. OK? (Y/N)

If user types N, the command is aborted and control returns to command prompt of the PROM programmer.

If the user enters Y, the system proceeds further. Any other character results in error message and repetition of the same prompt.

b) Now the following message appears:

Programming in progress...

The system proceeds with programming and verification on a byte by byte basis. Intelligent Programming Algorithm is used if the PROM can support it. This results in considerable reduction in programming time required for such devices.

c) If the complete programming is successful, the system will display a 16-bit checksum and control will return to the PROM programmer command fetch routine. The command prompt of PROM programmer is displayed on a new line.

If the programming is unsuccessful, the following information is displayed.

verify Error !!!

(PROM)=XX (Source) = XX
@ PROM address = XXXX

The displayed values indicate the location where programming was unsuccessful (X here indicates any hex digit)

NOTE: During programming from serial mode, the location in PROM and the corresponding data that is being programmed are displayed on the keyboard continuously. In case of failure, the address in PROM and corresponding PROM data, where the failure occurred are displayed.

10.3.2 V command

This command is used to verify the contents of a PROM against a source.

The parameters and their interpretation is completely similar to that of the P Command.

If the verification is successful, the checksum is displayed and programmer command prompt appears on the next line.

If the verification fails, a message and parameters at the point of failure are displayed as in the P command. Further the following prompt appears.

Verify Continue ? (Y/N)

If N is typed, control returns to start of PROM Programmer system
If Y is typed, verify operation continues with the next byte.

Thus the operation of this command is quite similar to the operation of the P command, except that here the PROM is just verified, not programmed.

10.3.3 B command

This command is used to check if a specified range in the PROM is blank (contains 0FFH)

This command requires the following three parameters:

Type	: Same as for P command
PROM Start	: The absolute starting address of the PROM
PROM End	: The absolute ending address of the PROM.

The parameters must satisfy the following relations:

i) PROM start <= Absolute last address of the PROM
ii) PROM End <= Absolute last address of the PROM and
>= PROM Start.

The parameter display and modification procedures are menu-driven and are similar to those of the P command

The PROM is checked for blank values in the specified range and if it is blank then the following message is displayed:

PROM is blank

Control returns to PROM Programmer System.

If a location in the specified range is not blank the following message is displayed.

PROM is not blank !!!

This message will be followed by the absolute PROM address of the first non-blank location as shown below:

First non-blank at address XXXX

(X is any hex digit)

Note that only the first non-blank location address is displayed. Subsequent locations may also be non-blank locations.

10.3.4 R Command

This command is used to transfer the contents of the PROM into the ESA 86/88-2 memory space.

This command requires the following four parameters:

Type	: Same as for P command
PROM Start	: Same as for B command
PROM End	: Same as for B command
Buffer Start	: Starting address in ESA 86/88-2 memory space. (can be given in segment : offset form or as offset only if default segment value is to be used).

The parameter display and modification procedures are menu driven and are completely similar to the ones described for P command.

The starting and ending addresses of the PROM must satisfy the relations described for the B (Blank check) command.

After the optional modification of the parameters, the contents of the PROM, in specified range are transferred into ESA 86/88-2 memory, starting at the specified Buffer starting address.

- ***IMPORTANT:***

The contents of PROM are transferred into successively alternate locations in ESA 86/88-2 memory.

During the transfer, as each byte is written into memory, it is read back and verified. If the write is successful for all the locations, a 16-bit checksum is displayed and control returns to the start of the PROM Programmer System.

If an error occurs during transfer (i.e unsuccessful write into a location), the following message is displayed.

Failure at memory location XXXX

(where **X** is any hex digit)

Control returns to the start of PROM Programmer system.

10.3.5 E command

This command is used to terminate the PROM Programming software and return control to the Serial Monitor. The sign-on message of the Monitor is again displayed followed by its command prompt on the next line.

10.4 OPERATION FROM KEYBOARD MONITOR

The Keyboard of ESA 86/88-2 has four keys which can be used to invoke the functions of the PROM Programming System. The keys and their functions are

Key	Function
PROG	Program a PROM
VRFY	Verify a PROM
BLNK	Blank check a PROM
TRFER	Read a PROM

It should be noted that the above function keys are redefinable by the user (Ref Chapter 3)

Following power on/reset, the system initializes these keys to point to appropriate routines in the PROM Programmer System. If the keys are subsequently redefined by the user, they must be restored to their default values for using the PROM Programming System.

Parameter Entry

The following information is common to all the commands.

Whenever a parameter entry is required, the system will display a message in the data field which indicates the type of parameter to be entered and it will display the default value in the address field with a dot. To retain this value user can press NEXT or EXEC, as required. Otherwise, user can enter the new value and then press NEXT or EXEC as required. All entries are made into the address field only. The different messages which can appear in the data field and their meanings are given below.

Message in the data field	**Meaning**
-typ	The type number of the PROM
-bSS	Segment value of the Buffer starting address
-bSO	Offset value of the Buffer starting address.
-bEO	Offset value of the Buffer ending address
-PrS	PROM starting address
-PrE	PROM ending address

All the parameter values are evaluated modulo 64K and a parameter entry is terminated by any valid delimiter. (Valid delimiters are: NEXT, PREV and EXEC). If an invalid entry is made, the message `-Err' is flashed in the address field and the prompt for the offending parameter is displayed again so that the user can enter the correct value.

Pressing the `NEXT' key takes the user to the next parameter if one is required - otherwise it is treated as error condition. Pressing the `PREV' key takes the user to the previous parameter, if one exists. Otherwise it is treated as error condition.

Pressing the `EXEC' key completes the parameter entry process of the command. Any parameters not explicitly displayed are assigned their default values and command execution continues with these values.

10.4.1 PROG Command:

a) This command is used to program a PROM.

This command requires the following four parameters in that order.

Type = PROM type (should be one of the types listed in section 10.1)

Buffer Start = Starting address of the source of data. It consists of a segment value and an offset value.

Buffer End = Ending address of the source of data. Only offset value is allowed.

PROM Start = Absolute starting address of the PROM (from where programming is to begin).

b) Note that these parameters must satisfy certain conditions as explained in the section 10.2.1 (P Command).

c) Once the correct parameter values are available, the system checks the PROM for blank values (0FFH) in the required zone. If the PROM is blank in the required zone, the system proceeds further. Otherwise, it displays the message FULL in the data field and waits for user input. Now if the user presses `EXEC' key, the system proceeds further. Any other key will return control to the monitor.

d) After this, the system proceeds with programming and verification on a byte by byte basis. Depending on the PROM type, it may take quite some time to program the entire PROM even though the system utilizes Intelligent Programming Algorithm wherever applicable to reduce the programming time significantly. As each location in the PROM gets programmed, the PROM address is displayed in the address field and the programmed data is displayed in the data field.

e) If the complete programming and verification is successful, the system will display a 16-bit checksum in the data field. Pressing any key returns control to the monitor.

If the programming is unsuccessful, the address of the failed PROM location is displayed in the address field and `FAIL' message is displayed in the data field. After a short delay, the data field will display the PROM data as most significant byte and the source data as least significant byte.

f) In either case (i.e after successful or unsuccessful programming), press any key to return control to the monitor.

10.4.2 VRFY COMMAND

a) This command is used to verify the contents of a PROM against a source.
b) The parameters and their interpretation is completely similar to that of the PROG command.
c) If the verification is successful, the 16-bit checksum is displayed in the data field. Now press any key to return control to the monitor.
d) If the verification fails, then the parameters at the failed location are displayed as in the PROG command. Now if `NEXT' key is pressed, verification continues with the next location. Any other key will terminate verify mode.
e) However, once again any key must be pressed to return control to the monitor.

10.4.3. BLNK COMMAND

a) This command is used to check if a specified range in the PROM is blank (contains 0FFH).

This command requires the following three parameters in that order:

type **:** Same as in PROG Command

PROM Start **:** The absolute starting address of the PROM

PROM End **:** The absolute ending address of the PROM

b) The parameters must satisfy certain relations as explained in the section 10.3.3 on B command.

c) Once valid parameter values are available, the PROM is checked for blank state in the specified range.

d) If the PROM is blank in the specified range, the message `PASS' is displayed in the data field. Otherwise, the address of the first non-blank location is displayed in the address field and the message `FAIL' is displayed in the data field.

e) In either case, press any key to return control to the monitor.

10.4.4 TRFER COMMAND

a) This command is used to transfer the contents of the PROM into the memory space of ESA 86/88-2.

This command requires the following four parameters in that order:

Type : Same as in PROG Command

PROM Start : Same as in BLANK command

PROM End : Same as in BLANK command

Buffer Start : Starting address in ESA 86/88-2 memory space.

b) These parameters must satisfy certain conditions as explained in the section 10.3.4 on R command.

c) Once correct parameter values are available, the system will read the PROM and transfer the contents to successively alternate locations starting from the specified buffer start.

d) If the transfer is successful, a 16-bit checksum is displayed. Otherwise the offset address of the offending location is displayed in the address field and the message `FAIL' is displayed in the data field.

e) In either case, press any key to return control to the monitor.

10.5 Examples

Assume that the PROM Programming module is installed as explained in the section 10.2

EXAMPLE: From keyboard, read the contents of 2732 PROM into memory locations 0:2000 to 0:3FFE.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	- E S A	8 6	;Assuming 8086 is installed.
TRFER	2 7 3 2.	- t y p	Transfer command type prompt
NEXT	0 0 0 0.	- P r S	PROM Start
NEXT	0 0 0 0.	- P r E	PROM End
F	0 0 0 F.	- P r E	
F	0 0 F F.	- P r E	
F	0 F F F.	- P r E	
NEXT	F F F F.	- B S S	Buffer start: segment
0	0 0 0 0.	- B S S	
NEXT	0 0 0 0.	- B S O	Buffer start offset
2	0 0 0 2	- B S O	
0	0 0 2 0	- B S O	
0	0 2 0 0	- B S O	
0	2 0 0 0	- B S O	
EXEC	1 2 3 4		Successful transfer
EXEC	- E S A	8 6	Return to monitor

NOTE:

Use Examine Byte or Examine word command to read the memory contents. Observe that the data is available in successive even locations ie 0:2000,2002,2004 etc.

EXAMPLE 2

From Serial monitor, program the contents of location 0:2000 to 0:3FFE into a 2764, starting AT 1000H. (User's entry is underlined only for clarity)

```
. P

ESA 86 / 88 PROM PROGRAMMER V1.0.
*(P/V/B/R/E) P
PROM Type  =      2732
Buffer Start  =   0000:2000
Buffer End    =   3FFE
PROM Start  =     0000

Any change of the above parameters ? (Y/N) Y
```

PROM type = 2732 - 2764
Buffer start = 0000: 2000-
Buffer End = 3FFE -
PROM start = 0000 - 1000

Programming in Progress ...
Checksum = 1724

- (P/R/V/B/E)

ESA 86/88/87 MONITOR Vx.y
.

CHAPTER 11

PARALLEL PRINTER INTERFACE

11.1 INTRODUCTION

ESA 86/88-2 provides the user the facility to obtain a hard copy via centronics compatible parallel printer interface. Any centronics compatible parallel printer could be connected to ESA 86/88-2 using this feature. However to get properly formatted listing specially during DI command it is advisable to use 80/132 column printer. This facility could be used only from serial mode of operation.

11.2 INSTALLATION AND OPERATION

To install the printer interface

- Switch OFF the power supply.
- Configure the system to serial mode of operation (SW4=OFF)
- Attach the printer (SW7 = ON) of on-board DIP SWITCH
- Connect the 26 PIN connector (PIN 26=GND) to J4 of ESA 86/88-2. The other end of connector must be connected to the printer.

NOTE:

1. The connector cable could be obtained from ESA as an option.
2. If the user decides to build one himself the connector requirements could be obtained from section 11.6 and 11.7. However note that the cable must be short enough such that it does not load the signals from 8255.

- Switch ON the power Supply. You should get the proper sign-on message.
- Now if the command prompt does not appear check the installation, the connector or Online/Ready indicator on the printer.
- When the command prompt appears, the printer is assumed to be “attached” to ESA 86/88-2.
- To "detach" the printer, put the switch SW7 of on-board DIP switch OFF and press RESET.

When the printer is attached, what ever character is output on to the console is also output to the printer. For example to obtain a hard copy of the contents of memory location in hex, (byte format), use the command D <addr1 >, <addr2> <CR>. The entire contents of memory locations between the specified range is printed exactly as it appears on the screen, including the command D <addr1>, <addr2> <CR>

NOTE : All control and invalid ASCII characters are printed as "."

Similarly to obtain the disassembled listing use the DI command. [after invoking the Assembler]

11.3 Theory of operation.

The printer software makes use of L0 8255 [at U27] connected to connector J4. This 8255 is intialized as given below, if the SW7 of DIP Switch is put ON.

PORT A	=	I/P
PORT B	=	O/P (data port)
PORT C	=	O/P

NOTE :

1) The user can use this 8255 as general purpose I/O, when not using the printer by putting OFF the SW7 of DIP Switch.

2) Handshaking is implemented using the STROBE* and BUSY signals

11.4 Error messages.

The following are the error messages, probable causes of occurrence and remedial measures.

If the Busy is not asserted the system will display the message " Printer Busy .... Waiting". Now, it keeps polling the status of Busy line, until it is asserted. This process can be aborted by pressing CNTL-C. The system will then display the message "Print aborted".

You can "detach" the printer by putting SW7 of the DIP switch off and then pressing the RESET key.

- If no acknowledgement for data transfer is received from printer then a message "No ACK from printer" is output. Please check the installation and connector once again in this case.

11.5 PARALLEL PRINTER PORT DETAILS

SPECIFICATIONS

- Centronics Compatible
- Synchronization via STROBE* pulses
- Handshaking via BUSY signal
- Logic levels TTL Compatible

11.6 CONNECTOR DETAILS

In all 11 lines are used as detailed below

PIN NO ON J4	SIGNAL	DIRECTION FROM ESA 86/88-2	DESCRIPTION	PIN NO ON CENTRONICS CONNECTOR
5 (PCO)	STROBE	O/P	STORBE* pulse to the printer	1
13(PB0)	Data 0	O/P	These signals	2
14(PB1)	Data 1	O/P	represent 8 bits	3
11(PB2)	Data 2	O/P	of parallel data	4
12(PB3)	Data 3	O/P	High = 1	5
9(PB4)	Data 4	O/P	Low = 0	6
10(PB5)	Data 5	O/P		7
7(PB6)	Data 6	O/P		8
8(PB7)	Data 7	O/P		9
16(PA7)	BUSY	I/P	A high indicates that Printer cannot receive data. The signal becomes high in following cases a) During data entry b) During Printing operation c) In OFF-LINE states d) During printer error status	11
26	GND		Signal ground	19

11.7 DIRECT OUTPUT TO PRINTER

As already described, when the printer interface is enabled, any character sent to the console is sent to the printer also. This facility is available in the serial mode of operation only. However, user can access a routine "Print String", to send a string directly to the printer. This routine can be called from the user's program when the system is operating in either of the two modes - keyboard or serial. Further, this routine prints the string independent of the setting of SW7. Thus this routine can be used to print the desired information even when the system is running in the keyboard mode. However, to use this routine, user program must first initialize the 8255 at U27 to mode 0 with port A as input and Ports B and C as outputs (mode control byte is 90H.) The details of the routine "Print String" are given below:

Name of the routine : Print String.

Function : Print a string. (The last character in the string must be ASCII NULL i.e. 00H.)

Calling Address : FC00 : 1B74 (user must access this routine through a for call ie opcode is 9AH.)

Input Parameters : ES:BX = Address of the string to be Printed.

Remarks : 1. User must save the necessary registers before making the far call and restore them afterwards.
: 2. The user program must first initialize the 8255 at U27.

: 3. The last character in the string to be printed must be 00H.

EXAMPLE : The following program prints the message "ESA". The program can be executed from the keyboard mode also. (NOTE: Ensure that the printer is connected to the system over the connector J4 and that it is in ONLINE mode.)

LOCATION	**CONTENTS**	**INSTRUCTION**	**COMMENTS**
0:2000	BA E6 FF	MOVW DX,#0FFE6	; Initialize 8255.
2003	B0 90	MOVB AL,#90	
2005	EE	OUTB DX	
2006	BA E4 FF	MOVW DX,#0FFE4	
2009	B0 01	MOVB AL,#01	
200B	EE	OUTB DX	
200C	B8 00 00	MOVW AX,#00	; Set up string address
200F	8E C0	MOVW ES,AX	
2011	BB 20 20	MOVW BX,#2020	
2014	9A 74 1B 00 FC	CALLS 1B74 0FC00	; Far call to print
2019	CC	INT 3	; Return to Monitor
...			
...			
...			
2020	0D 0A	DB 0D 0A	; String terminated
2022	45 53	DB 45 53	; by NULL
2024	41 0D	DB 41 0D	
2026	0A 00	DB 0A 00	

CHAPTER 12

COMMUNICATION WITH A HOST COMPUTER SYSTEM

12.1 INTRODUCTION

As already noted, ESA 86/88-2 operating in the serial mode, can be connected to either a CRT terminal or a host computer system. When a computer system is the controlling element, it must be executing a driver software to communicate with ESA 86/88-2.

XT86 is such an optional communication package which allows the user to establish a communication link between the ESA 86/88-2 trainer and a PC compatible computer system. The link is established between asynchronous serial ports of the computer (COM1/COM2) and ESA 86/88-2.

XT86 is supplied as a .EXE 'file on a 3½" DSHD diskette and can be executed on a PC compatible computer system under PC-DOS/MS-DOS operating system. A suitable RS 232 C cable has to be used for connecting ESA 86/88-2 to a PC/XT/AT compatible system.

XT86 fully supports all the commands of ESA 86/88-2. Further, it allows the contents of a disk file to be downloaded from the computer system into the memory of ESA 86/88-2. User can develop assembly language programs on the PC compatible computer system, cross-assemble them using a suitable cross-assembler to generate object code files and them use XT86 to download these object code files into ESA 86/88-2 for execution. Thus the extensive development facilities available on the PC compatible systems can be used to complement the facilities available on ESA 86/88-2.

Further XT86 allows uploading of data from the memory of ESA 86/88-2 to the disk file of the computer. Thus this facility can conveniently be used to save user programs.

12.2 INSTALLATION

NOTE: Make sure that you have made a back up copy of XT86.EXE before proceeding with the installation.

The detailed installation procedure is as follows:

a) Configure ESA 86/88-2 for serial mode of operation and set the serial port of ESA 86/88-2 for 9600 Baud and No parity (Refer sections 2.1.2 and 2.1.4).

b) Connect the computer system to ESA 86/88-2 over the COM1/COM2 serial port (Refer to Technical Manual of your system for details regarding the signal definitions on COM ports. The signal definitions of the RS 232C port of ESA 86/88-2 can be found in Appendix E.

c) Insert the diskette containing the file XT86.EXE into the available drive and run the program by typing

   XT86 or
   XT86/B to select Black and White mode if computer system has a CGA monitor.

d) Now the following message appears on the screen.

XT86 Version x.y

ELECTRO SYSTEMS ASSOCIATES PVT LTD

BANGALORE

Alt+S	**-**	**Set Communication Parameters**
Ctrl+F1	**-**	**Help**
Alt+F1	**-**	**Command Help**
<Esc>	**-**	**Clear Command**
<F1>	**-**	**Previous Command Character**
<F3>	**-**	**Command Recall**
Ctrl+U	**-**	**Upload Command**
Ctrl+D	**-**	**Download Command**
!Command	**-**	**DOS Shell/Command**
ALt+X	**-**	**Exit**

Press any key to Continue

e) XT86 checks for the presence of communication ports COM1 & COM2. If both ports are not available it will display the message

No serial port present as reported by BIOS

and exits to DOS. Otherwise XT86 will read the communication parameters from file "XT86.INS" and initializes the communication port. XT86 searches current directory for file "XT86.INS". If search fails, it will search the path given by the DOS environment variable INIT. If the file is not present, following message is displayed.

XT86.INS is not found!
Serial parameters are set to COM1, 9600, 8, 2, None
Do you want to change?

Yes No

If option "No" is selected the communication parameters: Serial Port COM1, Baud 9600, Data bits 8, Stop bits 2, Parity None are set. If option "Yes" is selected the communication parameters can be interactively modified as described in section 12.4.6.

Now XT86 attempts to establish communication between the computer and ESA 86/88-2. If successful, the command prompt "." Appears on the screen. (Subsequently during the power-on of RESET of the trainer, the sign-on message "ESA 86/88-2 MONITOR Vx.y" appears on the screen followed by command prompt "."). The word "E SE 86" will be displayed on the trainer's keyboard display. Otherwise it will display the message

Unable to transmit data

Retry or Ignore

If ESA 86/88-2 is not powered on, power it on and press <R> to retry to establish the communication. If the sign-on message does not appear, then "SErIAL" will appear on the trainer's address field indicating that trainer is trying to communicate with the host. The above mentioned message appears on the host side again. Pressing <I> will exit XT86 to DOS. Now check for the following.

A) Ensure that ESA 86/88-2 is connected to the correct COM port and that the COM port is functioning properly.

B) Ensure that ESA 86/88-2 is functioning properly and configured correctly.

C) Check the RS 232 C cable and its connections.

Since the communication package utilizes the hardware handshake signal DTR while communicating with ESA 86/88-2, the interfacing cable must support this signal also.

Note : XT86 utilizes an interrupt driven routine for reading characters from the COM port. Thus it is possible for XT86 to miss some characters if the system has any resident programs which are interrupt driven. (For example, many systems include a CLOCK program in the AUTOEXEC file, to display the time on the upper right corner of the screen.) hence it is desirable not to run any such resident programs while XT86 is running.

If the problem persists, please contact the manufacturer.

12.3 RETURNING TO DOS

User can terminate XT86 and return control to DOS by typing Alt+X when the program is waiting for a keyboard input.

12.4 OPERATIONAL DETAILS

The complete command set of the ESA 86/88-2 is transparent and is fully supported by XT86 (refer to chapter 4 for the serial monitor mode commands). Press Ctrl+F1 for help for command. In addition, XT86 supports the file download / upload and other commands which are explained below.

NOTE :
During parameter entry, the system expects the alphabetic characters to be in upper case. Thus it is convenient to use the key board with the CAPS LOCK on.

12.4.1 DOWNLOAD OPERATION:

This feature allows downloading of the contents of an object code file into the memory of ESA 86/88-2.

NOTE : The object code file must a "HEX" file with records in INTEL Extended HEX format. Please refer to the relevant INTEL manuals for the definition of INTEL Extended (16-Bit) HEX format. Most of the cross assemblers for 8086/8088 do produce object code files which are ".HEX" files with records in Intel Extended HEX format.

To perform download operation, type Ctrl+D in response to the command prompt (".").

The system will now prompt for the name of the disk file, from which the information is to be downloaded. The prompt is as follows :

Download filename [.HEX]:

Enter the file name with extension, terminated by <CR>. If the filename is invalid, it displays “File not found!” and prompts again for the filename. If the path specified is invalid, it displays a message “Path not found !" and prompts again for the filename. If none of the above errors occur, the system will read the file, gather the data in the specified address range, reformat the data for compatibility with the protocol required by ESA 86/88-2 and send the data to ESA 86/88-2.

Downloading in Progress.. XXXX.

After downloading is over, the system returns the command prompt of ESA 86/88-2. It also displays the starting address of each record being downloaded.

12.4.2 UPLOAD OPERATION:

This feature allows uploading of the data from the memory of ESA 86/88-2 to the computer system and saves the data in the specified disk file in INTEL 8-Bit HEX format.

To perform upload operation, type Ctrl+U in response to the command prompt (“.”). The system will now prompt for the name of the disk file, into which the information is to be uploaded. The prompt is as follows:

Upload filename [.HEX]

Enter the file name with extension, terminated by <CR>. If the file already exists, then the system will display

File already exists!
Overwrite?

Yes No Append

Select the first option by pressing <Y> to overwrite the contents of the existing file. Pressing <N> will let the user specify another file name. Select the third option <A> to append to the contents of the existing file.

If no error occurs, the system will prompt for the segment address as follows.

Segment Address :

The user has to enter segment from which the program has to uploaded. After this the system will prompt for starting address of the program as follows:

Start Address :

Enter the starting address terminated by <CR>. A maximum of four hex digits are allowed for the starting address.

Now the system will prompt for the ending address as follows:

End Address :

Enter the ending address terminated by <CR>. A maximum of four hex digits are allowed for the ending address.

After obtaining the filename, starting address and the ending address, the system will gather the data in the specified address range of the ESA 86/88-2, reformat the data into INTEL 8-Bit HEX records and store the data in the specified file.

Uploading in progress XXXX

While the uploading is going on, the system will display the starting address (XXXX) of each record being uploaded. Once the uploading is complete XT86 will let the user specify another address range. User may specify a new address range or enter <Esc> to terminate uploading operation.

The following error messages may appear during upload and download operations.

1. Invalid function number !

   This is XT86 internal error, therefore contact ESA 86/88-2 technical support for assistance.

2. File not found !

3. Path not found !

4. No more files !

5. Access denied !

6. Invalid file handle !

7. Insufficient Disk Space !

8. Unable to continue upload!

9. Colon is not present at the start of the Record !

10. Invalid data in (source file) the following Record!

11. Checksum Error in the following Record!

12.4.3 DOS COMMANDS :

At the command prompt ".", any valid DOS command can be entered preceded by "!". XT86 environment is saved and the DOS command is executed. Then XT86 environment is restored and XT86 command prompt is displayed again.

12.4.4 STATUS LINE:

During the session, XT86 displays many of the XT86 commands at the bottom line in reverse video for the convenience of user. The status line is displayed as

Ctrl+F1 Help, Alt+F1 CmdHelp, Alt+S Commn, <Esc> ClrCmd, Alt+X Exit, F1, F3,↑,↓

12.4.5 COMMAND RECALL:

This feature facilitates recall of the commands already entered by the user, upto a maximum of 16 commands. Press <F3> to recall the previous command. All the commands are kept in a circular buffer. User may use Up-arrow and Down-arrow keys to traverse through the sequence of commands in backward or forward direction respectively in a circular fashion. User may recall just the previous command, character by character, by pressing <F1> desired number of times. Current command being entered can be cleared by using <Esc> key any time before pressing <CR>.

12.4.6 COMMUNICATION:

Communication parameters can be set during the session by pressing Alt+S. List of parameters and their current values will appear on the screen. Select the desired parameter with the help of arrow keys and keep the space bar <SP> pressed till the desired value appears. The parameters allowed to be set are Communication Port (COM1/COM2), Baud Rate (110/150/300/600/1200/2400/4800/9600). and Parity bits (7/8), Number of Stop bits (1/2), and Parity (NONE/ODD/EVEN). After selecting the desired values press <CR> to set the parameters or press <Esc> to ignore the values.
Communication parameters can also be modified, while user is in DOS by editing the file XT86.INS. This file contains single data line, having five integers separated by blanks, representing various communication parameters, These five integers represent serial communication port, baud rate, number of data bits, number of stop bits and parity, in sequence. Table 12.1 shows details of the integer values and corresponding parameters.

Commn. Port	int1	Baud Rate	int2	Data Bits	int3	Stop Bits	int4	Parity	int5
COM1	0	110	0	7	0	1	0	odd	0
COM2	1	150	1	8	1	2	1	none	1
		300	2					even	2
		600	3						
		1200	4						
		2400	5						
		4800	6						
		9600	7						

12.4.7 HELP:

On-line help is available for all ESA 86/88-2 monitor commands and specific commands of XT86. Help facility can be selected by Ctrl+F1. A menu of commands is displayed from which desired command can be selected by using arrow keys and help information about that command is displayed in the remaining part of the screen. Context sensitive help is available using Alt+F1. This facility can be used if more information is desired about the command being entered against command prompt.

12.5 MORE ABOUT ".HEX" FILES

The files uploaded/downloaded under the control of XT86 are in the INTEL Extended Hex format which is described below:

12.5.1 Intel Extended HEX Format

In this format, binary values are coded in ASCII. For example, the binary value 0100 1111(=4FH) is coded as two ASCII characters "4" AND "F". In other words, the byte 4FH is represented by two bytes 34H (4 in ASCII) and 46H (F in ASCII). Thus this representation requires twice as many bytes as required by binary representation.

A file in extended Hex format can consist of upto four types for records. Each record begins with a record mark field containing 34H, the ASCII code for colon(:). This is followed by a RECORD LENGTH field which specifies the number of bytes of information which follow the RECORD TYPE field. The maximum number of record bytes allowed in the record is 255. After the data, the record has a CHECKSUM field that contains the ASCII representation of the two's complement of the eight-bit sum of the bytes that result from converting each pair of ASCII hexadecimal digits to one byte of binary, from and including data field. Therefore, the sum of all the ASCII pairs in a record after converting to binary, from the RECORD LENGTH field to and including the CHECKSUM field, is zero. The record ends with carriage Return, Line Feed sequence (0DH, 0AH).

The four types of records that may be present in a file of Extended Hex format are :

- Extended Address Record
- Start Address Record
- Data Record
- End of File Record

Extended Address Record:

RECD MARK	REC LEN	ZERO	REC TYPE	USBA	CHECK SUM
":"	"02"	"0000"	"02'	XXXX	XX

This record is used to specify the bits 19-4 of the Segment Base Address (SBA) where bits 3-0 are zero. Bits 19-4 of the SBA are referred to as the Upper Segment Base Address (USBA). Subsequent data bytes are loaded at the specified offsets relative to this USBA. The Extended Address Record may appear anywhere within the file. This value remains in effect until another Extended Address Record is encountered.

Example Record:

:02 0000 02 0200 FA

02+0000+02+0200+FA = 00 checksum validity

USBA = 0200H. For example, third byte in a subsequent data record with a load address of 0100H is loaded at 0200:0102.

Data Record:

RECE MARK	REC LEN	LOAD ADDRESS	REC TYPE	_.DATA._	CHECK SUM
":"	XX	XXXX	"00"		XX

The data record provides a set of ASCII characters that represent the hex digits of the data bytes. The method of loading these data bytes has already been described in the discussion of the Extended Address Record.

Example Record:

: 06 00 10 00 BA FF FF B0 40 90 B2

06+00+10+00+BA+FF+FF+B0+40+90+B2=00 checksum validity

Load address = 0010 & Number of data bytes =06

Assume the USBA currently valid is 0200. Then the data is to be loaded as shown below:

0200:0010=BA 0200:0011=FF 0200:0012=FF
0200:0013=B0 0200:0014=40 0200:0015=90

Start Address Record:

RECE MARK	REC LEN	LOAD ADDRESS	REC TYPE	CS	IP	CHECK SUM
":"	"04"	"0000"	"03"	XXXX	XXXX	XX

The start address record is used to specify the execution start address for the object file. This record can appear anywhere in the .HEX file. If the start address record is not present, CS and IP values will remain what they were before downloading.

Example Record:

: 04 00 00 03 02 00 01 00 F6

04 +00+ 00+ 03 +02 +00 +01 +00+ F6 = 00 checksum validity.

After loading, CS =0200 and IP= 0100

End of File Record :

RECD MARK	REC LEN	ZERO	REC TYPE	CHECK SUM
":"	"00"	"0000"	"01"	"FF"

This is a record of fixed structure and is used to specify the end of the .HEX file.

12.5.2 Using X8086

The most direct way of creating a file to be downloaded into ESA 86/88-2 is to use a cross-assembler for 8086/8088 which can generate the object code in Extended Hex format. X8086 is such a package, available from us as an option.

X8086 is a powerful cross-assembler for 8086/8088. It can run on any PC/XT/AT compatible system and supports all the standard mnemonics, pseudo-opcodes and addressing modes of 8086/8088. It includes a Linker that allows the generation of object code in Extended Hex file format. Thus user is urged to utilize X8086 for generating the HEX files to be downloaded into ESA 86/88-2. This procedure is the most direct and simplest one for exploiting the file download/upload features of the communication software XT86.

Note : The execution start address record generated by X8086 Linker does not follow the standard described already and hence is not supported by XT86. Thus user should not utilize this feature of X8086. In other words, the END statement in the source program should not have any operands.

Example : The distribution diskette includes a file DEMO6X.HEX which contains the object code for Example 6 of section 8.3. The code is in Extended Hex Format and the file was created using X8086 and the corresponding linker LINK.

The program is described, as already noted, in chapter 8. Download this DEMO6X.HEX file as already described (see section 12.4.1) and execute it from 0:2000H and observe the behavior to be as described in chapter 8.

The distribution diskette includes the source file also (DEMO6X.ASM). The steps involved in creating the .HEX file are:

i) Assemble DEMO6X.ASM using X8086 to create DEMO6X.OBJ

ii) Link the single file DEMO6X.OBJ

Specify code offset as 0 as the file is assembled from 0:2000H itself. Select E (for Extended Hex) as the option for output file format. This process creates DEMO6X.HEX which can then be downloaded under XT86.

As can be seen from the above description, the process of creating a .HEX file is extremely simple when X8086 is used.

12.5.3 Downloading .EXE files

If the source program in assembly language is assembled using the Macro Assembler MASM and then linked using the Microsoft Linker, the resulting object code file will have .EXE file structure. Such an .EXE file has two parts:

- Control and relocation information
- The load module itself.

This structure is designed for convenience in executing the files on a PC compatible computer system under PC DOS/MS DOS operating system (Please refer to the Technical Reference Manual of DOS for complete details regarding .EXE file structures). Thus to get absolute object code from .EXE files, one has to determine the load segment from where the load module is to be located and then relocate it appropriately using the control and relocation information present in the .EXE file. This process is accomplished by the LOADER program on the computer system. Because of these factors, downloading .EXE files into ESA 86/88-2 (an altogether different execution environment) is not a convenient procedure and user is encouraged to utilize X8086 cross-assembler.

However .EXE files with total size less than 64K can be downloaded as described in the following steps:

a) Convert the .EXE file to HEX file using the file converter package EXE2HEX.COM. This file is included on the distribution diskette. To invoke the file converter, type EXE2HEX followed by RETURN at the DOS prompt.

The system will now display the following sign-on message:

ELECTRO SYSTEMS ASSOCIATES PVT LTD

BANGALORE

EXE TO HEX FILE CONVERTER V1.0

Then the system will prompt for input file name as shown below:

FILE NAME =

Enter the file name with the extension followed by RETURN. If the specified filename is an invalid one (refer to your DOS manual for file name conventions), it displays the message "WHAT FILE SIR!" and returns to DOS. If the specified file does not exist, it displays the message "FILE NOT FOUND!" or the message "PATH NOT FOUND!" and returns to DOS. IF the file can not be read, it displays the message "ACCESS DENIED!" and returns to DOS. If a file already exists on the system with the same name as the input file name but with the extension of HEX, then it displays the message "HEX FILE ALREADY EXISTS!" and returns to DOS. If none of these errors occur, it creates an output file with the extension of .HEX. If it fails in this process, it will display the error message "TOO MANY OPEN FILES!" and returns to DOS. Otherwise the system will prompt for the LOAD SEGMENT value as shown below.

LOAD SEGMENT =

Enter the desired load segment value followed by Return.

The system will now prompt

ORIGIN VALUE =

Enter the origin value by RETURN.

NOTE : The advantage of specifying an origin value is as follows:

Assume that the source program has an origin statement like ORG 2000H (usually this will be the case because the user RAM in ESA 86/88-2 starts from 0:2000H. See the demo program DEMO6M.ASM). Now the .EXE file created by LINK does not start from 2000H!. Instead it starts from 0H and the contents of the file from 0H to 1FFFH will be all zeros. When such a file is converted to HEX file, the HEX file will have zeros in the address range 0 to 1FFFH. Downloading such a file will load zeros into the locations 0 to 1FFFH in ESA 86/88-2 memory space; an undesirable feature. The ORIGIN VALUE can be specified to avoid including this unnecessary data in the HEX file.

However if user wants to load from origin 0 (for example with a load segment value of 200H) , he/she can specify the ORIGIN VALUE to be 0.

Thus this feature allows user to delete if required, unwanted data from the HEX file.

Now the system will read the .EXE file and check if it is a valid .EXE file. If the specified input file does not have the .EXE file structure, the system displays the message “IMPROPER .EXE FILE!” and returns to DOS. If it is a valid .EXE file, then it will be relocated using the specified load segment value and the origin value, in accordance with the control and relocation information present in the .EXE file. The system will create an Extended Address Record using the load segment value and write this record on the output file. During relocation, if the system detects that the file size is greater than 64K, it displays the message “FILE TOO LARGE FOR CONVERSION!” and returns to DOS. If the relocation process is completed successfully, the system will write the relocated information on the output file as records in HEX format. Then it will obtain the CS and IP values from the .EXE file, relocate them and then create a start address record using the relocated CS and IP values. Then an end address record is generated and written to the output file. If the system fails in writing to the disk file, it displays the message “DISK WRITE FAILURE” and returns to DOS. After completing the process of writing, it closes the .HEX file. During the entire process, the hex records being created are displayed on the screen also. The system will then extract SS an SP values from the .EXE file, relocate them and then display them on the console. IF required, these values can later be used to set the SS and SP registers separately. Please note that DS an ES registers are supposed to point to an area called Program Segment Prefix when the .EXE file is given control under MS-DOS. However, when the conversion to HEX structure is performed for downloading the information into ESA 86/88-2, the Program Segment Prefix has no relevance and thus the system ignores these values.

b) After converting an .EXE file into a .HEX file as described above, user can install and run the XT86 and download the .HEX file into the memory of ESA 86/88-2 as per the procedures already described.

c) After downloading the file, user must set the DS, ES, SS and SP registers to the required values using the X command of the Serial Monitor of ESA 86/88-2. The SS and SP values displayed by EXE2HEX.COM can be made use of, if required.

d) Before executing the downloaded program, user is strongly urged to disassemble the code (or at least important parts of the code) using the on-board dissembler of ESA 86/88-2 and ensure that the program is correct and that the segment registers are set to proper values.

e) Execute the downloaded program.

Example : The distribution diskette includes a source program DEMO6M.ASM, which contains the source code for Example 6 of section 8.3 DEMO6M.HEX is the HEX file created according to the procedure described above. The steps are summarized below for the convenience of the user.

i) Use MASM to assemble DEMO6M.ASM. This process creates the file DEMO6M.OBJ

ii) Link the single file DEMO6M. OBJ using the linker LINK available on the system.

NOTE : The LINK program issues the warning "NO stack Segment" and reports that 1 error was detected. User can ignore this message, as the program really does not have to create a separate Stack Segment when running on ESA 86/88-2.

iii) Run EXE2HEX and specify DEMO6M.EXE as the input file. Specify a value of 0 for LOAD SEGMENT and a value of 2000 for the ORIGIN VALUE. Now the system creates DEMO6M.HEX file.

iv) Connect ESA 86/88-2 to the system and run XT86.

V) Download the DEMO6M. HEX file and execute it. Observe the behavior of this program to be as described in chapter 8.

12.5.4 Downloading .COM Files

Assume that an .EXE file has been created using MASM and LINK as described in the previous section. Such a file can be converted to a .COM file using the standard DOS command EXE2BIN (Refer the DOS user's manual for details regarding this command and for limitations on the structure of .EXE file which are to be converted to .COM files).

A. COM program contains the pure memory of the program and has the following distinct characteristics:

a) The size of the code and data parts of the file is less than 64K.

b) There is no separate STACK segment

c) No segment fixups are necessary as the file is assumed to be segment relocatable.

d) CS, DS, ES and SS are assumed to contain the same value when the program begins execution.

This structure is designed for executing small programs, in a convenient way on PC/XT/AT compatible computer systems, under PC DOS/MS DOS. However, a .COM file can be downloaded into ESA 86/88-2 as described below:

a) Convert the .COM file to .HEX file using the file converter package COM2HEX.COM which is included on the distribution diskette.

To invoke the file converter, type COM2HEX followed by RETURN at the DOS prompt. The system will now display the following sign-on message:

Then the system will prompt for the input file name, load segment value and origin value in a manner exactly similar to the one described in the previous section. The meaning and interpretation of these parameters and the possible error messages are also same as in the previous section. However, as a .COM file is not expected to have a separate stack segment COM2HEX does not display SS and SP values as is done by EXE2HEX program.

Example : The distribution diskette includes a source program DEMO6C.ASM, which contains the source code for example 6 of section 8.3. DEMO6C.HEX is the HEX file created according to the procedure described above. These steps are summarized below for the convenience of the user.

i) Use MASM to assemble DEMO6C.ASM. This process creates the file DEMO6C.OBJ.

ii) Link the single file DEMO6C.OBJ, using the linker LINK available on the system. This process creates the DEMO6C.EXE file.

NOTE : The LINK program issues the warning “No Stack Segment” and reports that 1 Error was detected. User can ignore this message, as .COM files never have a stack Segment.

iii) Convert the .EXE file to .COM file using the standard DOS command EXE2BIN.

(The command sequence is EXE2BIN DEMO6C.EXE DEMO6C.COM)

iv) Run COM2HEX.COM and specify DEMO6C.COM as the input file. Specify a value of 0 for LOAD SEGMENT and a value of 2000 for the origin value. This process creates DEMO6C.HEX file.

V) Connect ESA 86/88-2 86/88-2 to the system and run XT86.

Vi) Download DEMO6C.HEX file and execute it. Observe the behavior of this program as described in chapter 8.

12.6 DEMO FILES

The distribution diskette includes .HEX files containing the object codes for all the example programs described in chapter 8 (Programming Examples). User can download these files and execute the programs according to the directions given in chapter 8. The correspondence between the .HEX files and example programs is given below:

.HEX file	Example program
a) DEMO821.HEX	Example 1 of section 8.2
b) DEMO822.HEX	Example 2 of section 8.2
c) DEMO823.HEX	Example 3 of section 8.2
d) DEMO824.HEX	Example 4 of section 8.2
e) DEMO831.HEX	Example 1 of section 8.3
f) FHELPX.HEX	Example 2 of section 8.3
g) DEMO833.HEX	Example 3 of section 8.3
h) DEMO834.HEX	Example 4 of section 8.3
i) DEMO835.HEX	Example 5 of section 8.3
j) DEMO6X.HEX DEMO6M.HEX DEMO6C.HEX	Example 6 of section 8.3
k) DEMONDP1.HEX	Example 1 of section 8.4
l) DEMONDP2.HEX	Example 2 of section 8.4
m) PICDEMO.HEX	Example of section 8.5

Note that the distribution diskette includes the following source programs also.

1. DEMO6X.ASM Source program for Example 6 of section 8.3. It's use is described in section 12.5.2

2. DEMO6M.ASM Source program for the same example i.e., Example 6 of section 8.3. It's use is described in section 12.5.3.

3. DEMO6C.ASM Source program for Example 2 of section 8.3, Must be assembled using X8086 as described in section 12.5.2 to create FHELPX.HEX.

4. FHELPX.ASMSource program for Example 2 of section 8.3. Must be assembled using X8086 as described in section 12.5.2 to create FHELPX.HEX.

5. DEMONDP2.ASM Source program for Example 2 of section 8.4. Must be assembled using MASM with R option. DEMONDP2.HEX is created according to the procedures described in section 12.5.3.

for After - Sales Service and Spares

please contact our authorised service centre :

Integrated Services and Consultancy

Second Floor, #37, "EMBEDDED HOME"
36th Cross, II Block, Rajajinagar
BANGALORE - 560 010. India
Fax : 91-80-3325615 Phone : 91-80-3521165
e-mail : esaindia@vsnl.com

ELECTRO SYSTEMS ASSOCIATES PVT LTD

4215 J K Complex First Main Road Subramanyanagar
P.O. Box : 2139 BANGALORE - 560 021 INDIA
Fax : 91 80 3325615 Telephone : 3323029 3322924
e-mail : esaindia@vsnl.com